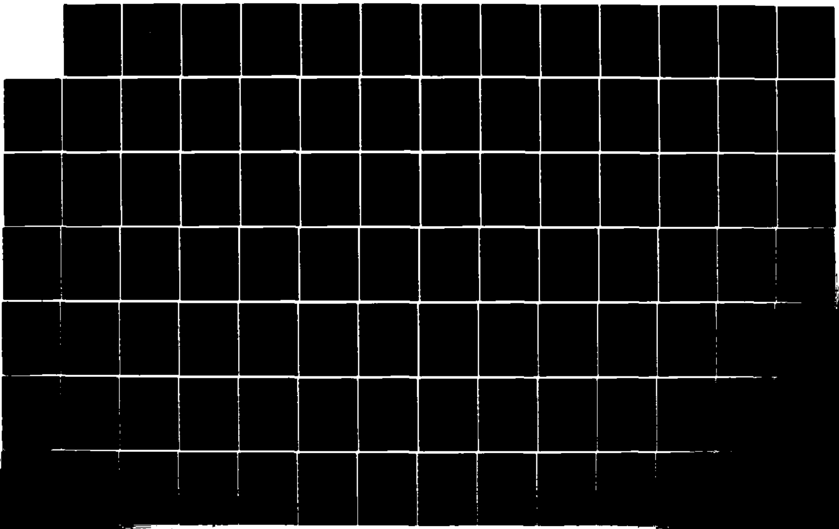
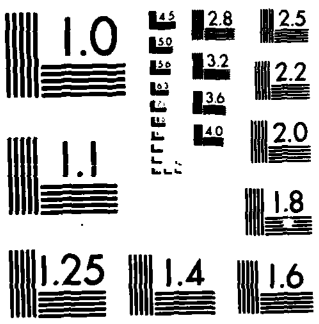
AD-A143 690 DEVELOPMENT OF REAL-TIME ERROR ELLIPSES AS AN INDICATOR 1/2
OF KALMAN FILTER PERFORMANCE(U) NAVAL POSTGRADUATE
SCHOOL MONTEREY CA J JAROS MAR 84

UNCLASSIFIED F/G 9/2 NL




1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

2

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DEVELOPMENT OF REAL-TIME ERROR ELLIPSES
AS AN INDICATOR OF KALMAN
FILTER PERFORMANCE

by

Joseph Jaros

March 1984

Thesis Advisor: A. Gerba, Jr.

Approved for public release, distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Development of Real-Time Error Ellipses as an Indicator of Kalman Filter Performance		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; March 1984
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Joseph Jaros		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		12. REPORT DATE March 1984
		13. NUMBER OF PAGES 119
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Error Ellipsoids; Kalman Filter; Extended Kalman Filter

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An error ellipse plotting routine was developed to provide real-time indication of Kalman filter performance. The study included an evaluation of the Hewlett-Packard HP-86 computer system's capability for providing real-time tracking information and an evaluation of the computer's possible use on the three-dimensional underwater tracking range at the Naval Underwater Weapons Engineering Station, Keyport, Washington. A series of tracking —

runs were used to demonstrate both linear and extended Kalman filtering. Information obtained from the error ellipses was used to modify filter parameters for improved filter performance. It was found that the error ellipse was useful as a tool for indicating filter performance and for making decisions regarding filter parameter modification. The HP-86 provided accurate, reliable results and it could be used for on-line graphics. However, the computing speed of the HP-86 computer as used in this study was too slow for on-line processing of the three-dimensional tracking problem.

Development of Real-Time Error Ellipses as an Indicator of Kalman Filter Performance

by

Joseph Jaros
Commander, United States Navy
B.S., University of Texas, 1967

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1984

Author: [illegible]

Approved by: Alex Gerba Jr.
Thesis Advisor

H A Titus
Second Reader

[illegible]
Chairman, Department of Electrical and Computer Engineering

[illegible]
Dean of Science and Engineering

ABSTRACT

An error ellipse plotting routine was developed to provide real-time indication of Kalman filter performance. The study included an evaluation of the Hewlett-Packard HP-86 computer system's capability for providing real-time tracking information and an evaluation of the computer's possible use on the three-dimensional underwater tracking range at the Naval Underwater Weapons Engineering Station, Keyport, Washington. A series of tracking runs were used to demonstrate both linear and extended Kalman filtering. Information obtained from the error ellipses was used to modify filter parameters for improved filter performance. It was found that the error ellipse was useful as a tool for indicating filter performance and for making decisions regarding filter parameter modification. The HP-86 provided accurate, reliable results and it could be used for on-line graphics. However, the computing speed fo the HP-86 computer as used in this study was too slow for on-line processing of the three-dimensional tracking problem.

TABLE OF CONTENTS

I. INTRODUCTION

The Kalman filter's importance as an estimator and predictor is well documented. Providing real-time information concerning filter performance so that on-line adjustments to filter parameters can be made continues to be an area of high interest. This study investigates the usefulness of the error ellipse as a tool for providing a real-time indication of filter performance.

Part of the investigation involves an evaluation of the Hewlett-Packard HP-86 computer system's capacity to operate in a real time tracking environment, and its capabilities for providing information concerning filter performance. The rationale behind this investigation is based on the requirement for the Naval Underwater Weapons Engineering Station, Keyport, Washington, to accurately acoustically track torpedoes on a three-dimensional underwater tracking range. A knowledge of the range operation is not within the scope of this study. It is sufficient to know that presently the range receives four time measurements every 1.31 seconds, and these measurements are nonlinear functions of the torpedo position.

To gain a better understanding of the error ellipse, a 4-state tracking scenario was chosen for this study.

Initially, the linear tracking problem is discussed, followed by an investigation of the nonlinear problem. Of primary interest are on-line methods to improve filter performance using information provided by the error ellipse for filter parameter modification.

II. KALMAN FILTER THEORY

A. LINEAR MATHEMATICAL MODEL

1. The Plant

For this model the state and measurement equations for the plant are linear. Hence the discrete form is used. The assumed plant model is described by a linear, vector difference equation:

$$\underline{x}(k+1) = \underline{\phi x}(k) + \underline{\Delta u}(k) + \underline{\Gamma w}(k) \quad \text{(State Equation)} \qquad (2\text{-}1)$$

and a linear, vector measurement equation:

$$\underline{z}(k) = \underline{cx}(k) + \underline{v}(k) \qquad (2\text{-}2)$$

where:

$\underline{x}(k)$	is an n-dimensional column vector, denoting the state of the plant at "time" k.
$\underline{u}(k)$	is the deterministic control input, an m-vector, at time k.
$\underline{w}(k)$	is a p-dimensional vector representing any random forcing inputs at time k.
$\underline{z}(k)$	is a q-dimensional vector representing measurements made at time k.
$\underline{v}(k)$	is a q-dimensional vector representing random measurement made at time k.

$\underline{\phi}$, $\underline{\Delta}$, $\underline{\Gamma}$, and $\underline{c}$ are assumed constant coefficient matrices of dimension nxn, nxm, nxp, and qxn respectively.

2. Noise Processes

In order to place probabilistic structure on the noise processes $\underline{v}(k)$ and $\underline{w}(k)$ the following assumptions are made:

(a) $\underline{v}(k)$ and $\underline{w}(k)$ are individually white processes, that is, for any k and l, with $k \neq l$, $\underline{v}(k)$ and $\underline{v}(l)$ are independent random variables, and $\underline{w}(k)$ and $\underline{w}(l)$ are independent random variables.

(b) $\underline{v}(k)$ and $\underline{w}(k)$ are individually zero mean, Gaussian processes with known covariances.

(c) $\underline{v}(k)$ and $\underline{w}(k)$ are independent processes.

Thus for the measurement noise:

$$\text{MEAN:} \quad E[\underline{v}(k)] = \underline{0} \quad (k=0,1,2,3,\ldots) \tag{2-3}$$

$$\begin{aligned} \text{COVARIANCE:} \quad E[\underline{v}(k)\underline{v}^T(l)] &= E[\underline{v}(k)]E[\underline{v}^T(l)] \\ &= 0 \qquad k \neq l \\ &\overset{\Delta}{=} \underset{\sim}{R}_k \qquad k = l \end{aligned}$$

$$\text{or } E[\underline{v}(k)\underline{v}^T(l)] = \underset{\sim}{R}_k \delta_{kl} \quad (k,l=0,1,2,\ldots) \tag{2-4}$$

where δ_{kl} is the Kronecker delta function defined as:

$$\delta_{kl} \begin{cases} = 0, & k \neq l \\ = 1, & k = l \end{cases}$$

Likewise for the random forcing input:

$$E[\underline{w}(k)] = \underline{0} \qquad (k=0,1,2,3,\ldots) \qquad (2\text{-}5)$$

$$E[\underline{w}(k)\underline{w}^T(l)] \triangleq \underline{Q}_k \delta_{kl} \qquad k,l=0,1,2,3,\ldots) \qquad (2\text{-}6)$$

$\underline{Q}_k$ and $\underline{R}_k$ are nonnegative definite symmetric for all k. Also since $\underline{v}(k)$ and $\underline{w}(k)$ are zero mean and independent then:

$$E[\underline{v}(k)\underline{w}^T(l)] = \underline{0} \qquad (2\text{-}7)$$

For the purposes of this study, unless otherwise specified $\underline{Q}_k$ and $\underline{R}_k$ are considered to be known and constant, although both may be time varying.

3. Initial State Description

For the initial state of the difference equation (2-1) it is unlikely that $\underline{x}_o$ will be available. Hence, it is assumed that $\underline{x}_o$ is a Gaussian random variable of known mean $\overline{\underline{x}}_o$ and known covariance $\underline{P}_o$, i.e.,

$$E[\underline{x}(0)] = \overline{\underline{x}}_o$$

$$E\{[(\underline{x}_o - \overline{\underline{x}}_o)][(\underline{x}_o - \overline{\underline{x}}_o)]^T\} = \underline{P}_o$$

This choice for the initial state has the advantage of causing the subsequent estimation scheme to be unbiased for all τ. [Ref. 1] Further it is assumed that the initial state and the measurement noise are uncorrelated:

$$E[\underline{x}(0)\underline{v}^T(k)] = E[\underline{v}(k)\underline{x}^T(0)] = \underline{0} \quad (k=0,1,2,3,\ldots)$$

Also the initial state and the random forcing input are uncorrelated:

$$E[\underline{x}(0)\underline{w}^T(k)] = E[\underline{w}(k)\underline{x}^T(0)] = \underline{0} \quad (k=0,1,2,3,\ldots)$$

B. DISCRETE-TIME ESTIMATION

1. The Estimator Equations

The estimation problem involves generating an optimal estimate for $\underline{x}(j)$ for the system described by the difference equation (2-1) from the noisy measurements $\underline{z}(0),\underline{z}(1),\ldots,\underline{z}(j)$. This estimate will be denoted by $\hat{\underline{x}}(j/j)$, which means the estimate of $\underline{x}$ at time j given measurements at times up to and including time j. The estimate must be optimal in the sense that the expected value of the sum of the squares of the error in the estimate is a minimum, i.e.:

$$E\{[(\hat{\underline{x}}(k/k) - \underline{x}(k)]^T[\hat{\underline{x}}(k/k) - \underline{x}(k)]\} = \text{minimum}$$

The estimator is characterized by the linear relationship:

$$\hat{\underline{x}}(k/k) = \hat{\underline{x}}(k/k-1) + \underline{G}(k)[\underline{z}(k) - \underline{c}\hat{\underline{x}}(k/k-1)] \quad (k=0,1,2,..) \tag{2-8}$$

where

$\hat{\underline{x}}(k/k)$	is the optimal (minimum variance) estimate of $\underline{x}(k)$ given observations at times up to and including k.
$\hat{\underline{x}}(k/k-1)$	is the optimal one-step prediction of $\underline{x}(k)$ given observations at times up to and including k-1.
$\underline{G}(k)$	is the optimal estimation gain matrix which will minimize the variance of estimation error.

For the initial estimate $\hat{\underline{x}}(0/0)$, the estimator equation (2-8) is initialized with $\hat{\underline{x}}(0/-1)$, which is not a random variable. If $\hat{\underline{x}}(0/-1)$ is selected such that:

$$\hat{\underline{x}}(0/-1) = E[\underline{x}(0)] = \bar{\underline{x}}_o$$

it can be shown that this choice of $\hat{\underline{x}}(0/-1)$ makes the estimator unbiased for all k. [Ref. 1] The estimator's best available information concerning $\underline{x}(k-1)$ is the estimate $\hat{\underline{x}}(k-1/k-1)$, therefore it is reasonable to assume that

$$\hat{\underline{x}}(k/k-1) = \underline{\phi x}(k-1/k-1) + \underline{\Delta u}(k-1) \tag{2-9}$$

is the best prediction.

In summary, equations (2-8) and (2-9) are the estimator equations, with $\underline{\hat{x}}(0/-1) = \underline{\bar{x}}_o$ as the initial condition.

2. Gain and Covariance Equations

Without going into detailed derivations, the optimal estimator gains, $\underline{G}(k)$, used in the estimator equation (2-8), are those which satisfy:

$$\underline{G}(k) = \underline{P}(k/k-1)\underline{C}^T[\underline{C}\underline{P}(k/k-1)\underline{C}^T + \underline{R}]^{-1} \tag{2-10}$$

$$\underline{P}(k/k) = [\underline{I} - \underline{G}(k)\underline{C}]\underline{P}(k/k-1) \tag{2-11}$$

$$\underline{P}(k+1/k) = \underline{\phi}\underline{P}(k/k)\phi^T + \underline{Q} \tag{2-12}$$

with the initial conditions:

$$\underline{P}(0/-1) \triangleq \underline{P}_o = E\{[\underline{x}(0) - \underline{\bar{x}}_o][\underline{x}(0) - \underline{\bar{x}}_o]^T\}$$

where

$$\underline{P}(k/k) = E\{[\underline{\hat{x}}(k/k) - \underline{x}(k)][\underline{\hat{x}}(k/k) - \underline{x}(k)]^T\}$$

is the covariance of estimation error matrix.

$$\underline{P}(k/k-1) = E\{[\hat{\underline{x}}(k/k-1) - \underline{x}(k)][\hat{\underline{x}}(k/k-1) - \underline{x}(k)]^T\}$$

is the covariance of one-step prediction error matrix.

$$\underline{Q} = E[\underline{\Gamma}(k)\cdot\underline{w}(k)\cdot\underline{w}^T(k)\cdot\underline{\Gamma}^T(k)]$$

is the state excitation matrix.

$\underline{P}(k/k)$ and $\underline{P}(k/k-1)$ are symmetric, positive definite matrices.

Several observations can be made concerning the linear Kalman gain (2-10), covariance (2-11, 2-12) and estimator (2-8) equations.

(a) The estimator gains, $\underline{G}(k)$, do not depend on the measurement data and hence can be precomputed, stored, and used as the processing measurements become available.

(b) Although not obvious from the equation, the time-varying gain, $\underline{G}(k)$, depends in time as:

$$\underline{G}(k) = \frac{1}{(k+1)} \qquad \text{[Ref. 2]} \qquad (2\text{-}13)$$

Thus the effect is to weight the correction term, $[\underline{z}(k) - \underline{cx}(k/k-1)]$, in the estimator equation (2-8) less heavily as time progresses. The advantage of a greater initial weight allows for possibly large differences between $\underline{z}(k)$ and $\underline{x}(k/k-1)$ during the initial observations, and a large gain will result in a significant change in the next estimate. This advantage is also borne out in that there is less confidence in the quality of the estimates during the early observations compared with the quality after numerous observations. Hence the later an observation, the less drastic an estimate will be altered or affected by an isolated observation discrepancy.

(c) In general, the variance of estimation error decreases in a manner analogous to the gain schedule (2-13), i.e., it decreases as k grown larger, reflecting greater confidence in the estimate as the number of observations increases. Selection of the proper initial condition, $\underline{P}_o$, is important when studying the effect of measurement errors on the behavior of the estimate. So $\underline{P}_o$ should be assigned pessimistic values which would correspond to a lack of information about the initial state. In cases when the initial state is completely unknown, then $\underline{P}_o \to \infty I$. [Ref. 3]

(d) The $\underline{Q}$ matrix serves to compensate for model errors and prevents the covariance matrix from becoming too small or optimistic. A small covariance matrix would result in a small filter gain, and subsequent observations are essentially ignored, which could result in filter divergence. The $\underline{Q}$ matrix prevents $\underline{G}(k)$ from approaching zero by adding uncertainty to the system which is reflected in a degradation of certainty (increase in $\underline{P}(k+1/k)$).

C. NONLINEAR ESTIMATION - EXTENDED KALMAN FILTER

In many practical applications, the state equations and/or measurement equations are nonlinear. Before the Kalman filter equations can be used, the problem must be linearized and the Kalman filter equations are applied with some modification.

1. Nonlinear Model

Consider a nonlinear discrete system of state and observation equations given by:

$$\underline{x}(k+1) = \underline{f}(\underline{x}(k), \underline{u}(k), k) + \underline{w}(k) \tag{2-14}$$

and

$$\underline{z}(k) = \underline{h}(\underline{x}(k), k) + \underline{v}(k) \tag{2-15}$$

In these equations $\underline{f}$ and $\underline{h}$ are nonlinear functions of the state variable $\underline{x}$, $\underline{w}(k)$ is the plant excitation noise, and $\underline{v}(k)$ is the measurement noise. The plant noise and measurement noise are assumed to be uncorrelated, zero-mean, and white. The same equations (2-3 thru 2-7) apply as for the linear model.

2. Extended Kalman Filter Equations

In order to apply the linear filter equations, equations (2-14) and (2-15) are expanded about the best estimate of the state at that time and only the first-order terms are kept.

That is, defining A(k) as:

$$\underline{A}(k) = \left.\frac{\partial \underline{f}}{\partial \underline{x}}\right| (\hat{\underline{x}}(k/k), \underline{u}(k), k)$$

and

$$\underline{H}(k) = \left.\frac{\partial \underline{h}}{\partial \underline{x}}\right| (\hat{\underline{x}}(k/k-1))$$

As can be seen from the above equations, the filter estimates, $\hat{\underline{x}}(k/k)$ and $\hat{\underline{x}}(k/k-1)$ are used as the "best" estimates about which the linearization is performed. The matrices $\underline{A}(k)$ and $\underline{H}(k)$ must be used to generate $\underline{G}(k)$ so it is available to process $\underline{z}(k)$ when it is obtained. The modified extended Kalman filter equations are then:

Gain Equation:

$$\underline{G}(k) = \underline{P}(k/k-1)\underline{H}^T (k)[\underline{H}(k) \cdot \underline{P}(k/k-1) \cdot \underline{H}^T (k) + R]^{-1} \tag{2-16}$$

Filter Update Equation:

$$\hat{\underline{x}}(k/k) = \hat{\underline{x}}(k/k-1) + \underline{G}(k)[\underline{z}(k) - \underline{h}(\hat{x}(k/k-1))] \tag{2-17}$$

Prediction Equation:

$$\hat{\underline{x}}(k+1/k) = \underline{f}(\hat{\underline{x}}(k/k),\ \underline{u}(k),\ k) \qquad (2\text{-}18)$$

Covariance of Estimation Error Equations:

$$\underline{P}(k/k-1) = \underline{A}(k-1)\underline{P}(k-1/k-1)\underline{A}^T(k-1) + \underline{Q}(k-1) \qquad (2\text{-}19)$$

$$\underline{P}(k/k) = [\underline{I} - \underline{G}(k)\underline{H}(k)]\underline{P}(k/k-1) \qquad (2\text{-}20)$$

For the initial estimate $\hat{\underline{x}}(0/0)$, equation (2-17) is initialized with $\hat{\underline{x}}(0/-1)$ with

$$\hat{\underline{x}}(0/-1) = E[\underline{x}(0)] = \bar{\underline{x}}_o$$

$\hat{\underline{x}}(0/-1)$ is also used to initially evaluate $\underline{H}(k)$. As in the linear case:

$$\underline{P}(0/-1) = \underline{P}_o = E[(\underline{x}_o - \bar{\underline{x}}_o)][(\underline{x}_o - \bar{\underline{x}}_o)^T]$$

III. ERROR ELLIPSOIDS

A. THEORY

Since the estimate $\hat{\underline{x}}(k/k)$ is unbiased in the Kalman filter equations, the $\underline{P}(k/k)$ matrix represents the covariance of the error in the estimate. If the estimate were biased, $\underline{P}(k/k)$ would represent the second-moment matrix rather than the covariance matrix. Hence, $\underline{P}(k/k)$ provides significant information about the accuracy of the estimate. If the physical model is accurately described by the state and measurement equations (2-1, 2-2), then $\underline{P}(k/k)$ can be used to describe the manner in which the estimate converges (or diverges) to the true state. Examination of the $\underline{P}(k/k)$ matrix directly, element by element, is not a realistic approach, since the matrix contains n^2 elements, where n is the number of state variables. To simplify the situation the concept of the error ellipsoid is used. [Ref. 4]

As discussed earlier, the assumptions are made that the initial state of the plant $\underline{x}_o$ is Gaussian, as are the random processes $\underline{v}(k)$ and $\underline{w}(k)$. Using these assumptions it follows that $\underline{x}(k)$ and $\hat{\underline{x}}(k/k)$ are also Gaussian since they are linear combinations of Gaussian variables and deterministic quantities. Using the same rationale, the estimation error, defined as:

$$\underline{e}(k/k) \triangleq \hat{\underline{x}}(k/k) - \underline{x}(k)$$

is also Gaussian. Using the fact that the mean of the estimation error is 0, the probability density function for $\underline{e}(k/k)$ is:

$$p_e[\underline{e}(k/k)] = [(2\pi)^{n/2}|\underline{P}(k/k)|^{1/2}]^{-1}$$

$$\exp[-1/2\underline{e}^T (k/k)\underline{P}^{-1}(k/k)\underline{e}(k/k)] \qquad (3\text{-}1)$$

The density function, $p_e[\underline{e}(k/k)]$ will have a constant value whenever the exponent has a constant value. That is:

$$-1/2\underline{e}^T (k/k)\underline{P}^{-1} (k/k)\underline{e}(k/k) = c$$

or

$$\underline{e}^T (k/k)\underline{P}^{-1}(k/k)\underline{e}(k/k) = c^2 \qquad (3\text{-}2)$$

where c is an arbitrary constant.

As demonstrated by Sorenson [Ref. 1] and Kirk [Ref. 2], it can be shown that the locus of points $\underline{e}(k/k)$ which satisfy equation (3-2) are hyperellipsoids. For the two-dimensional case which is of concern, equation (3-2) describes an ellipse. This can be seen by fixing time and rewriting (3-2) as:

$$\underline{e}^T \underline{we} = c^2 \qquad (3\text{-}3)$$

where

$\underline{w} = \underline{P}^{-1}$ (k/k) (a 2 x 2 symmetric matrix)

Expanding the left side of (3-3) gives:

$$w_{11}e_1^2 + w_{12}w_{21}e_1e_2 + w_{22}e_2^2 = c^2$$

which because of symmetry gives:

$$w_{11}e_1^2 + 2w_{12}e_1e_2 + w_{22}e_2^2 = c \qquad (3\text{-}4)$$

Since $w_{11} > 0$, $w_{22} > 0$, and $w_{11}\ w_{22} > w_{12}^2$, equation (3-4) describes an ellipse, in which the principal axes do not coincide with the coordinate axes. The ellipse is rewritten in terms of $\underline{y}(1)$ and $\underline{y}(2)$ as the coordinate axis, and it can be shown that $\underline{y}(1)$ and $\underline{y}(2)$ are the eigenvectors of $\underline{w}$ with λ_1 and λ_2 defined as the corresponding eigenvalues. [Ref. 2] (See Figure 3-1). The equation for the ellipse can be rewritten in terms of a coordinate system having unit vectors in the directions of $\underline{y}(1)$ and $\underline{y}(2)$ as the basis vectors. The ellipse equation becomes:

$$\lambda_1\underline{y}^2(1) + \lambda_2\underline{y}^2(2) = c^2 \qquad (3\text{-}5)$$

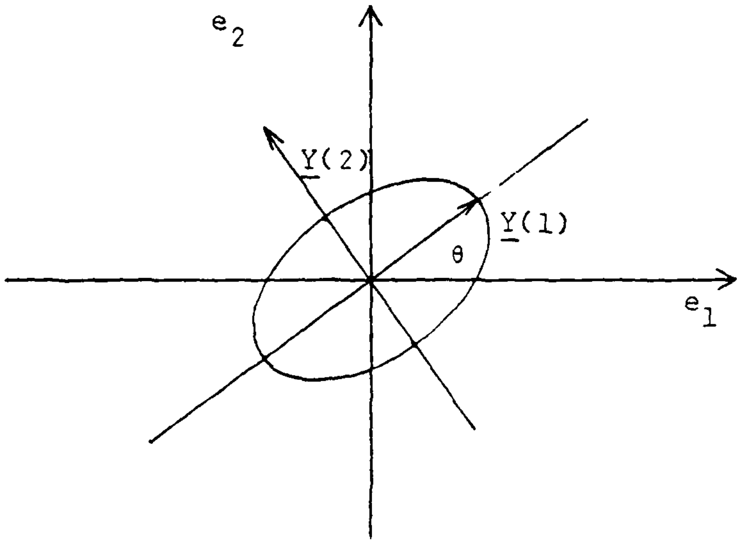


Figure 3-1 Error Ellipse

Remembering that $\underline{w} = P^{-1}(k/k)$, it can be shown that the corresponding eigenvectors and eigenvalues for $\underline{w}^{-1}=\underline{P}(k/k)$ are $\underline{y}(1)$, $\underline{y}(2)$, α_1, and α_2, where $\alpha_1 = \frac{1}{\lambda_1}$ and $\alpha_2 = \frac{1}{\lambda_2}$. Equation (3-5) can be rewritten:

$$\frac{\underline{y}^2(1)}{\alpha_1} + \frac{\underline{y}^2(2)}{\alpha_2} = c^2 \qquad (3-6)$$

For the purposes of this study, the term error ellipsoid refers to the specific case when c = 1. So equation (3-6) becomes:

$$\frac{\underline{y}^2(1)}{\alpha_1} + \frac{\underline{y}^2(2)}{\alpha_2} = 1$$

In terms of the Cartesian coordinates x', y', which use e_1 and e_2 as basis vectors:

$$x' = x\cos\theta + y\sin\theta$$

$$y' = -x\sin\theta + y\cos\theta$$

where θ is the angle of rotation of the axes and can be computed from:

$$\theta = 1/2 \tan^{-1}\left[\frac{2\mathrm{cov}(e_x e_y)}{\mathrm{var}(e_x) - \mathrm{var}(e_y)} \right] \qquad \text{[Ref. 5]}$$

For a given value of c it is possible to integrate the probability density over the surface of the error ellipse to obtain the probability that a particular sample point will lie within the ellipsoid. For this study, n = 2, c = 1, and the probability the error is inside the ellipse is 0.394.

In summary, the error ellipsoid can be used to characterize the concentration of the estimate about the true value of the state. A decrease in the magnitude of an axis of the ellipse is an indication that the error in the estimate is decreasing in that direction.

One important item that needs to be pointed out is that often the components of the state vector represent entirely different types of variables, for example the components might represent range, velocity, and depth. Since the two-dimensional ellipses are determined by using two components of the state vector, it is reasonable to examine submatrices relating state variables of the same character. Doing so will preclude most scaling difficulties when plotting the ellipses, and provide more meaningful insight in to the results.

B. ERROR ELLIPSOIDS AND FILTER DIVERGENCE

Thus far the discussion has centered around using submatrices of the $\underline{P}(k/k)$ covariance of estimation error matrix as the input for the error ellipse to indicate filter performance. As proposed by Heffes [Ref. 6] and Nishimura [Ref. 7], $\underline{P}(k/k)$ can be considered as a "design" covariance matrix $\underline{P}^d$, using the assumption that the only errors are in $\underline{Q}_k$, $\underline{R}_k$, and $\underline{P}_o$ with the following inequalities holding for all k:

$$\underline{Q}_k^d \geq \underline{Q}_k^a \ , \ \underline{R}_k^d \geq \underline{R}_k^a \quad \underline{P}_o^d \geq \underline{P}_o^a \tag{3-7}$$

with the subscript "d" indicating designed and "a" indicating actual. Equation (3-7) implies more input noise, more measurement noise, and more initial state uncertainty in the design than actually exists. This conservative filter design results in a somewhat pessimistic design error covariance $\underline{P}^d(k/k)$. The actual error covariance $\underline{P}^a(k/k)$ resulting from using a filter designed with $\underline{Q}^d$, $\underline{R}^d$, and $\underline{P}_o^d$ is related to the design error covariance in the following manner:

$$\underline{P}^d(k/k) \geq \underline{P}^a(k/k)$$

This result is particularly useful when one simply does not know accurately the noise covariance of the input or output, but an upper bound is known. Designing assuming the noise covariance is at its upper bound will result in $\underline{P}^a(k/k)$ being upper bounded by $\underline{P}^d(k/k)$. In some sense a worst case design results. Filter divergence exists when the design error covariance $\underline{P}^d(k/k)$ remains bounded while the error performance matrix $\underline{P}^a(k/k)$ becomes very large relative to $\underline{P}^d(k/k)$ or is, in fact, unbounded.

IV. PROBLEM DEFINITION

A. PROBLEM DESIGN

The purpose of the tracking problem is to study the use of error ellipsoids as real-time indicators of filter performance. In order to keep the design model realistic albeit reasonably simplified for ease of study, a two-dimensional tracking problem using several different tracks has been selected.

1. Linear Tracking

All tracks are based on an x-y coordinate system with the target moving in the x or y direction relative to the sensor located at the origin. Thus for aircraft tracking, altitude is considered constant, as is depth for torpedo tracking.

Defining the state variables as:

$x_1 = x$	x-coordinate of the target location.
$x_2 = \dot{x}$	velocity of target (v_x) in x-direction.
$x_3 = y$	y-coordinate of the target location.
$x_4 = \dot{y}$	velocity of target (v_y) in y-direction.

resulting in a state vector:

$$\underset{\sim}{x} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} \qquad (4\text{-}1)$$

The following are the state equations:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= w_1(t) \\ \dot{x}_3(t) &= x_4(t) \\ \dot{x}_4(t) &= w_2(t) \end{aligned} \qquad (4\text{-}2)$$

where $w_1(t)$ and $w_2(t)$ are assumed to be uncorrelated, random processes that account for unknown target accelerations and nonlinear target motions. Writing the discrete form of the state equations gives:

$$x_1(k+1) = x_1(k) + T \cdot x_2(k) + \frac{T^2}{2} \cdot w_1(k)$$

$$x_2(k+1) = x_2(k) + T \cdot w_1(k) \qquad (4\text{-}3)$$

$$x_3(k+1) = x_3(k) + T \cdot x_4(k) + \frac{T^2}{2} \cdot w_2(k)$$

$$x_4(k+1) = x_4(k) + T \cdot w_2(k)$$

or

$$\underline{x}(k+1) = \underline{\phi x}(k) + \underline{\Gamma w}(k) \qquad (4\text{-}4)$$

With T, the sampling period equal to 1 second, the matrices $\underline{\phi}$ and $\underline{\Gamma}$ are:

$$\underline{\phi} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \underline{\Gamma} = \begin{bmatrix} .5 & 0 \\ 1 & 0 \\ 0 & .5 \\ 0 & 1 \end{bmatrix} \qquad (4\text{-}5)$$

It is assumed that the sensor gives noisy, but uncorrelated measurements of x and y. Hence the discrete measurement equations are:

$$z_1(k) = x_1(k) + v_1(k)$$
$$z_2(k) = x_3(k) + v_2(k) \qquad (4\text{-}6)$$

with $v_1(k)$ and $v_2(k)$ uncorrelated random noise.
Thus for the measurement equation:

$$\underline{z}(k) = \underline{cx}(k) + \underline{v}(k) \qquad (4\text{-}7)$$

the matrix c is:

$$\underline{c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

For the initial run and unless otherwise noted the following values for the Gaussian random processes will be used:

$$E[\underline{v}(k)] = \underline{0} \text{ for all } k$$

$$E[\underline{v}(k)\underline{v}^T(k)] = \begin{bmatrix} 20\text{x}10^3 & 0 \\ 0 & 20\text{x}10^3 \end{bmatrix} m^2 = \underline{R} \text{ for all } k$$

$$\underline{\sigma}_v = \begin{bmatrix} 150 \\ \\ 150 \end{bmatrix} \text{ m} = \text{the standard deviation of measurement noise.}$$

$$E[\underline{w}(k)] = \underline{0} \text{ for all } k$$

$$E[\underline{w}(k)\underline{w}^T(k)] = \begin{bmatrix} 100 & 0 \\ \\ 0 & 100 \end{bmatrix} (\text{m/sec}^2)^2 = \text{cov } w \text{ for all } k$$

$$\underline{\sigma}_w = \begin{bmatrix} 10 \\ \\ 10 \end{bmatrix} \text{ m/sec}^2 = \text{the standard deviations of the random forcing input.}$$

The covariance of estimation error matrix is initialized:

$$\underline{P}_o = \underline{P}(0/-1) = \begin{bmatrix} 10^2 & 0 & 0 & 0 \\ 0 & 10^2 & 0 & 0 \\ 0 & 0 & 10^2 & 0 \\ 0 & 0 & 0 & 10^2 \end{bmatrix}$$

Since the filter is to be unbiased, the initialization:

$$\hat{\underline{x}}(0/-1) = \bar{\underline{x}}_o = \text{Initial condition of the problem.}$$

2. Nonlinear Tracking

The state equations are the same as for the linear tracking problem. The measurement equation is considered as a noisy range measurement by the tracking sensor and is characterized as:

$$z(k) = [x_1^2(k) + x_3^2(k)]^{1/2} + v_1(k) \qquad (4-8)$$

Thus z(k) is a nonlinear function of the states. Using equation (2-14) and with T = 1 second:

$$\underline{f}(\underline{x}(k), \underline{u}(k), k) = \begin{bmatrix} x_1(k) + x_2(k) \\ x_2(k) \\ x_3(k) + x_4(k) \\ x_4(k) \end{bmatrix}$$

Taking partial derivatives of $\underline{f}$ with respect to $\underline{x}$ gives:

$$\underline{A}(k) = \left.\frac{\partial \underline{f}}{\partial \underline{x}}\right|_{(\hat{\underline{x}}(k/k),\ \underline{u}(k),\ k)} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \underline{\Phi}$$

using equation (2-15):

$$\underline{h}(\underline{x}(k),\ k) = [x_1^2(k) + x_3^2(k)]^{1/2}$$

and taking the partial derivative with respect to $\underline{x}$ gives:

$$\underline{H}(k) = \left.\frac{\partial \underline{h}}{\partial \underline{x}}\right|_{(\underline{x}(k/k-1))} = \left.\left[\frac{x_1(k)}{[x_1^2(k)+x_3^2(k)]^{1/2}},\ 0,\ \frac{x_3(k)}{[x_1^2(k)+x_3^2(k)]^{1/2}},\ 0 \right]\right|_{\hat{\underline{x}}(k/k-1)}$$

Using the above results with the same values for the random noise processes as previously stated for the linear case, the extended Kalman filter equations can now be applied to the nonlinear tracking problem.

B. COMPUTER SIMULATION

1. Computer Hardware/Software

a. Hardware

All computer simulations were run on the Hewlett-Packard HP-86 personal computer. This particular model was chosen to evaluate its capabilities in determining its usefulness in actual torpedo tracking at the underwater tracking range at Naval Underwater Weapons Engineering Station, Keyport, Washington. The HP-86 system used included keyboard, 9 inch CRT monitor connected through an integrated monitor interface, and two HP Flexible Disc Drives connected through an integrated disc interface. Plotting was done on a HP-7225B Plotter and printing on a HP-2631G Printer. These peripherals were interfaced using a HP-IB Interface module. Because the system uses interface select codes, the HP-IB factory preset code was set at 7, which is the select code for the printer/disc interface. This code however did not work when interfacing with the external plotter and printer, since duplicate select codes are not allowed. So the internally set select code of the HP-IB was set to 8 for proper system operation.

b. Software

The HP-86 has 60K built-in, useable bytes of computer memory, expandable to 572K using either 32K, 64K, or 128K Memory Modules. A HP-86 plug-in ROM was required to operate the external plotter. Also a Matrix ROM was used to reduce program length and computer run time.

All programs were written in BASIC programming language using REAL (full) precision, which provides 15 digit precision. Appendix B provides an explanation of the program options and Appendix C contains the program listings used for this study.

2. Track Generation

To evaluate the real-time use of the error ellipse as an indicator of filter performance, Monte Carlo simulation runs were made. Four tracks were generated by separate programs and one second incremental values of x, y, v_x, and v_y were stored in data files. Appendix I contains an explanation of the generation of tracks three and four.

3. Noise Generation

In order to simulate the random noise processes, the computer's random number generator was used and the generated numbers scaled accordingly. For each track and each different value of noise sigma, a different generator "seed number" was used. These noise values produced were added to the applicable true track values to simulate a sensor measurement corrupted by independent Gaussian noise. For all cases where filter parameters were varied for a particular track under a specific noise condition, one noisy track was generated, stored, and used throughout that particular simulation. This was done for ease of filter performance comparison.

4. Gating Scheme

In order to preclude catastrophic filter failure due to excessive measurement noise, a bound was established for the maximum acceptable limits of measurement noise. A three-sigma gate was designed using the covariance of the measurement noise, $\underline{R}$, and the predicted covariance of error matrix $\underline{P}(k/k-1)$. The gate is defined as:

$$\text{Gate}(k) = 3(p_{ii_{max}}(k/k-1) + R_{ii})^{1/2}$$

This gate is the maximum error allowable for the measurement at time k. If the absolute difference between the actual measurement received and the predicted measurement is greater than the three-sigma gate, then that particular measurement data is rejected as unacceptable. When this occurs, the filter gain, $\underline{G}(k)$, is set to zero, resulting in that measurement being ignored and the prediction of the states set equal to the estimate, that is:

$$\underline{x}(k/k) = \underline{x}(k/k+1)$$

5. <u>Collection of Statistics</u>

In order to study error ellipses as an indicator of filter performance, statistics were calculated, on line, after each measurement during the Monte Carlo run. The statistics computed were relative error (in some cases the error was normalized), error mean, error variance, and error covariance for the positional variables. The following equations apply:

$$\text{Relative Error} = \underline{e}(k/k) = \underline{x}(k) - \underline{\hat{x}}(k/k)$$

(filter error residual)

$$\text{Error Mean} = \underline{\bar{e}}(k/k) - 1/k \sum_{j=1}^{k} \underline{e}(j/j)$$

$$\text{Error Variance} = \text{Var}[e_i(k/k)] = 1/k \sum_{j=1}^{k} [e_i(j/j)]-[\bar{e}_i(k/k)]^2$$

$$i = x_1,\ x_2,\ x_3,\ x_4$$

$$\begin{array}{l} \text{Positional} \\ \text{Error} \\ \text{Covariance} \\ \text{Matrix} \end{array} = \begin{bmatrix} \text{Var}[e_{x_1}(k/k)] & \begin{array}{l} 1/k \sum_{j=1}^{k} [e_{x_1}(j/j)][e_{x_3}(j/j)] \\ -[\bar{e}_{x_1}(k/k)\cdot\bar{e}_{x_3}(k/k)] \end{array} \\ \begin{array}{l} 1/k \sum_{j=1}^{k} [e_{x_1}(j/j)][e_{x_3}(j/j)] \\ -[\bar{e}_{x_1}(k/k)\cdot\bar{e}_{x_3}(k/k)] \end{array} & \text{Var}[e_{x_3}(k/k)] \end{bmatrix}$$

V. TARGET TRACKING AND ERROR ELLIPSE ANALYSIS

A. LINEAR TRACKING

Track 1 depicts a target approaching at a constant velocity of 223.6 feet per second. The solid line of Figure 5.1 indicates the true track of the target and the numbers along the track indicate the time in seconds. The target was tracked in a measurement noise, $\underline{\sigma}_v$ = 150, with the random forcing noise $\underline{\sigma}_w$ = 1. $\underline{R}$ was set for 20,000. The dots indicate the filtered track using the linear Kalman filter equations. Figures 5.2 and 5.3 are the filter error ellipses for this run, computed at increments of 10 seconds. As can be seen the ellipse size decreases with increasing time indicating filter convergence. The computed ellipse surface areas shown on the figures confirm this. Figure 5.4 shows the filtered track for the case where $\underline{\sigma}_v$ has been increased to 300 and all other parameters remain the same. The error ellipses of Figure 5.5 computed for 10-second increments show increasing area indicating filter divergence. Figure 5.6 shows the error ellipses for the same track run but this time the ellipses were computed using a "statistics window" of 10. By this is meant that the ellipses were derived from statistics computed for the last 10 data measurements. All previous data is disregarded. Using this method, the ellipses of Figure 5.6 show filter convergence from iterations 15 to 25. The filtered track of Figure 5.4 confirms this. Figure 5.7 shows the results for the same track parameters, except in this case a statistics window of 5 was used. The window 5 ellipse area at time 25 (5179 sq ft) is much less than the area of either the run with the statistic window of 10 (11,540 sq ft) or the run with no window at all (65,500 sq ft). This is expected since the filter is essentially "locked on" at time 20, and the window 5 ellipse at time 25 disregarde all data previous to time 20. Figure 5.8 shows the error ellipses for the same track but the measurement noise was increased to $\underline{\sigma}_v$ = 400, while $\underline{R}$ was kept at 20,000. The error ellipses indicate filter divergence, and indeed the filtered track headed off in the wrong direction.

Track 2 depicts a target approaching at a constant speed of 500 feet per second in the -y direction. Figure 5.9 depicts the solid line track and the dots indicate the linear filtered track with $\underline{\sigma}_v$ = 150, $\underline{\sigma}_w$ = 1 and $\underline{R}$ = 20,000. Figure 5.10 and 5.11 are the error ellipses for the run. No statistics window was used. Other than the fact that the ellipse area is decreasing, the shape of the ellipse provides little additional information. Figure 5.12 and 5.13 show the ellipses for the normalized error. With the same measurement noise sigma for both the x-position and y-position measurements, the normalized error ellipse's shape and orientation reflect the target track proximity to either axis. In Figure 5.12 the ellipse major axis indicates a large normalized error in the x-direction. This is to be expected since the target maintains a constant x-position of 500 feet, while the y-position is initially 20,000 feet. However, near time 40 as the target approaches the x-axis, the normalized y error increases and becomes so large at the x-axis crossing that the normalized x error becomes insignificant in comparison. This can be seen in Figure 5.13 for the ellipses at time 45 and 55 compared with the ellipse at time 35. The normalized error ellipse is an excellent indicator of target proximity to an axis, but the rapid shifts in ellipse surface area make it difficult to determine filter convergence.

Track 3 depicts a target approaching on a parabolic track at a speed of 200 feet per second. Figure 5.14 is the true track, with a linearly filtered track indicated by the dots. For this run $\underline{\sigma}_v$=150, $\underline{\sigma}_w$=10, and $\underline{R}$=20,000. Figure 5.15 are the error ellipse plots for times 40, 50, and 60, the period of the highest rate of change in x- and y-velocity. The ellipse areas increase with time indicating divergence. Figure 5.16 and 5.17 are the ellipse plots using a 10-data point and a 5-data point statistics window respectively. In both cases the ellipse areas for time 60 are less than time 50 indicating the filter has tracked around the curve.

Using error ellipses based on a statistics window in this tracking situation provides a better indicator of filter performance.

For the second run of track 3, $\underline{\sigma}_y$ was increased to 300 and all other parameters remained the same. Figure 5.18 shows the resulting filtered track, which obviously didn't track around the curve. With the large amount of measurement noise and considering that the maximum random forcing input, i.e. the maximum acceleration in the x- and y-direction, occurs between times 40 and 60, it is a logical place to lose track. Another factor to be considered is the decrease in gain as k increases. By time 40 the gains have little influence. So a system was incorporated in the program to "reinitialize" the filter by setting the gains to G(0), if certain conditions were met. After several trial and error runs, it was determined that if a statistics window of 10 were used, and the gains reinitialized if the error ellipse area increased consecutively a certain number of times, that the filtered track would follow around the curve. Figure 5.19 is the filtered track using a statistics window of 10 and reinitializing the filter if the error ellipse area increased consecutively during five 1-second increments. The error ellipses for that run are shown in Figures 5.20 and 5.21. As indicated on the figures, the filter was reinitialized four times and the ellipse areas for the 10-second increments increased, until reinitializing the filter at time 52 locked the filter in. Consequently the error ellipse areas for time 60 and 70 decreased, indicating convergence.

In the final run, the filter was reinitialized after 10 consecutive 1-second error ellipse increases, with all other parameters remaining the same. Figure 5.22 is the filtered track; Figures 5.23 and 5.24 are the error ellipse plots for this run. As indicated on Figure 5.24 the filter was reinitialized only once at time 57. A comparison of ellipse areas for the last two runs shows that, with the exception of time 70, the areas were larger in the first run when the filter was reinitialized 4 times. This is expected since reinitializing results in larger gains producing more widely vary estimates, and hence greater error variance. At time 70 the error ellipse area of the first run is less since in the first run the last filter reinitialization occurred at time 52 versus time 57 in the second run. The first filtered track had more time to settle out by time 70, resulting in less error.

B. NONLINEAR TRACKING

Track 4 depicts a target moving at a constant velocity of 50 feet per second (30 kts) in the x-direction for 15 seconds, at which time the target turns and travels in the -y-direction. (See Figure 5.25) Using the extended Kalman filter with $\underline{\sigma}_v$=30 and $\underline{R}$=900, a series of tracking runs were made for various values of COVW (σ_w^2=from 20 to 200). In none of these runs did the filter successfully track the target around the turn. Figure 5.26 shows the results for the case when COVW=150. In this instance the filter lost track as the target came out of the turn. Trying to track through a turn using a constant COVW (and hence, a constant $\underline{Q}$) did not work. So a scheme was devised to vary COVW dependent on information derived from the error ellipse. After several trial runs for this particular track, it was determined that if the error ellipse area increased consecutively for 7 iterations of k, COVW would be doubled, and if the ellipse area decreased consecutively for 5 iterations of k, COVW would be halved. Initially the the trial runs were made without a statistics window, and the filter did not track successfully. Without the statistics window, the old data weighted down the statistics, and the error ellipses were not indicative of what was currently happening. So it was decided to use a statistics window. Windows of 5, 10, and 15 were tried. Window 5 proved to be too responsive and window 15 not responsive enough. So a statistics window of 10 was chosen for the tracking run. With $\underline{P}_o=10^2$, $\underline{\sigma}_v=30$, $\underline{R}=900$, and COVW initially set at 20, the tracking run was made. Figure 5.27 depicts the filtered track output, and Figures 5.28-5.30 are the 10-second incremental error ellipses for the run. As can be seen, the filter did track around the curve. Figure 5.29 shows the ellipse areas are becoming less between times 55 and 65. Also indicated below the plots are the values of

COVW for the k time the plot was computed. During this particular run COVW varied from an initial value of 20 up to 160, and then decreased to 40 by the end of the run.

Using the same filter parameters as above except $\underline{\sigma}_v$ was increased to 200, and $\underline{R}$ to 40,000, another run was made. The filter did not track at all. During this run COVW varied from an initial value of 20 up to 40 and decreased to 5 by the end of the run. Obviously, the criteria for increasing and decreasing COVW was not effective. More trial runs were necessary to determine the optimum consecutive increases or decreases of the ellipse areas before adjusting COVW accordingly.

A second approach to the nonlinear tracking problem, one that was used earlier for the linear case, is to reinitialize the filter if certain conditions are met. Again, using trail and error runs with and without statistics windows, it was determined that using a statistics window of 10 gave the best results. Using initial conditions of COVW=150, $\underline{P}_o=10^2$, $\underline{\sigma}_v=30$, and $\underline{R}=900$, several runs were made, reinitializing the filter if the error ellipse area increased consecutively for a certain number of iterations of k. Of the runs attempted, the best results were obtained when the filter was reinitialized if the area increased for 5 consecutive iterations of k. Figure 5.31 is the filtered track for this run, and Figures 5.32-5.34 are the error ellipse plots. Indicated below the plots are the times, k, when the filter was reinitialized. For this particular run the filter was reinitialized 5 times. It should be noted that when reinitializing the filter, $\underline{P}(k/k-1)$ was reset to $10^2 \times \underline{P}_o$. $\underline{P}_o$ was not large enough to be effective in getting the filter back on track. The initial value of 10^2 was used for $\underline{P}_o$ to reflect the high confidence in the initial state conditions. Other values of $\underline{P}_o$ did not work as well.

Using the same parameters as above except the filter was reinitialized after 7 consecutive error ellipse area increases, another run was made with the resulting track depicted in Figure 5.35. A comparison with Figure 5.31 reveals that using 7 area increases as the criterion for filter reinitialization resulted in poorer filter performance, as can be shown from the error ellipses.

The final nonlinear filter tracking runs attempted involved simultaneously varying COVW and filter reinitialization. The results were disastrous, and highly unpredictable. It was an interesting experiment in futility, and no meaningful results could be obtained.

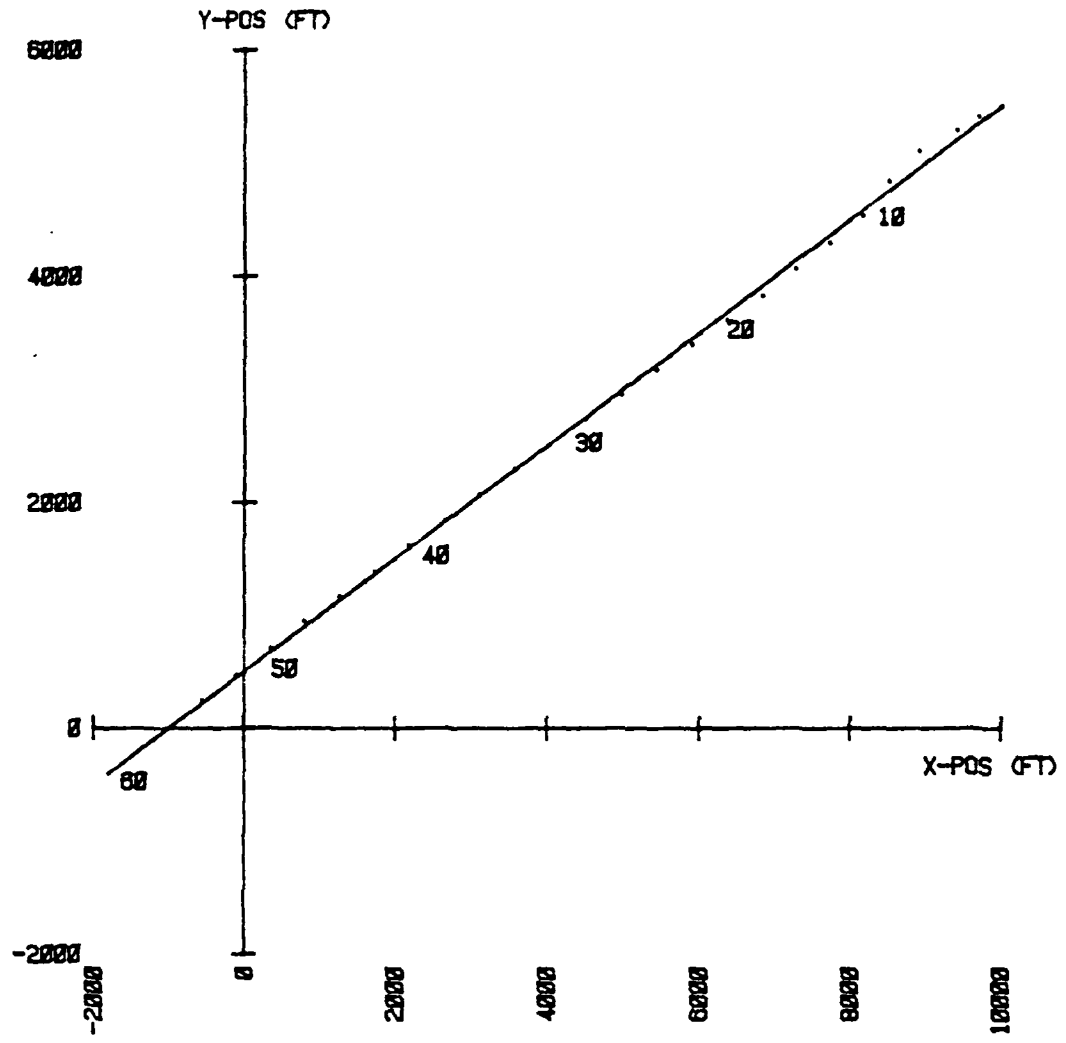


Figure 5.1 Solid Line Track 1, Vel 223.6 ft/sec; Dots Indicate Filtered Track, $\underline{\sigma}_v$=150, $\underline{\sigma}_w$=1, $\underline{R}$=20,000

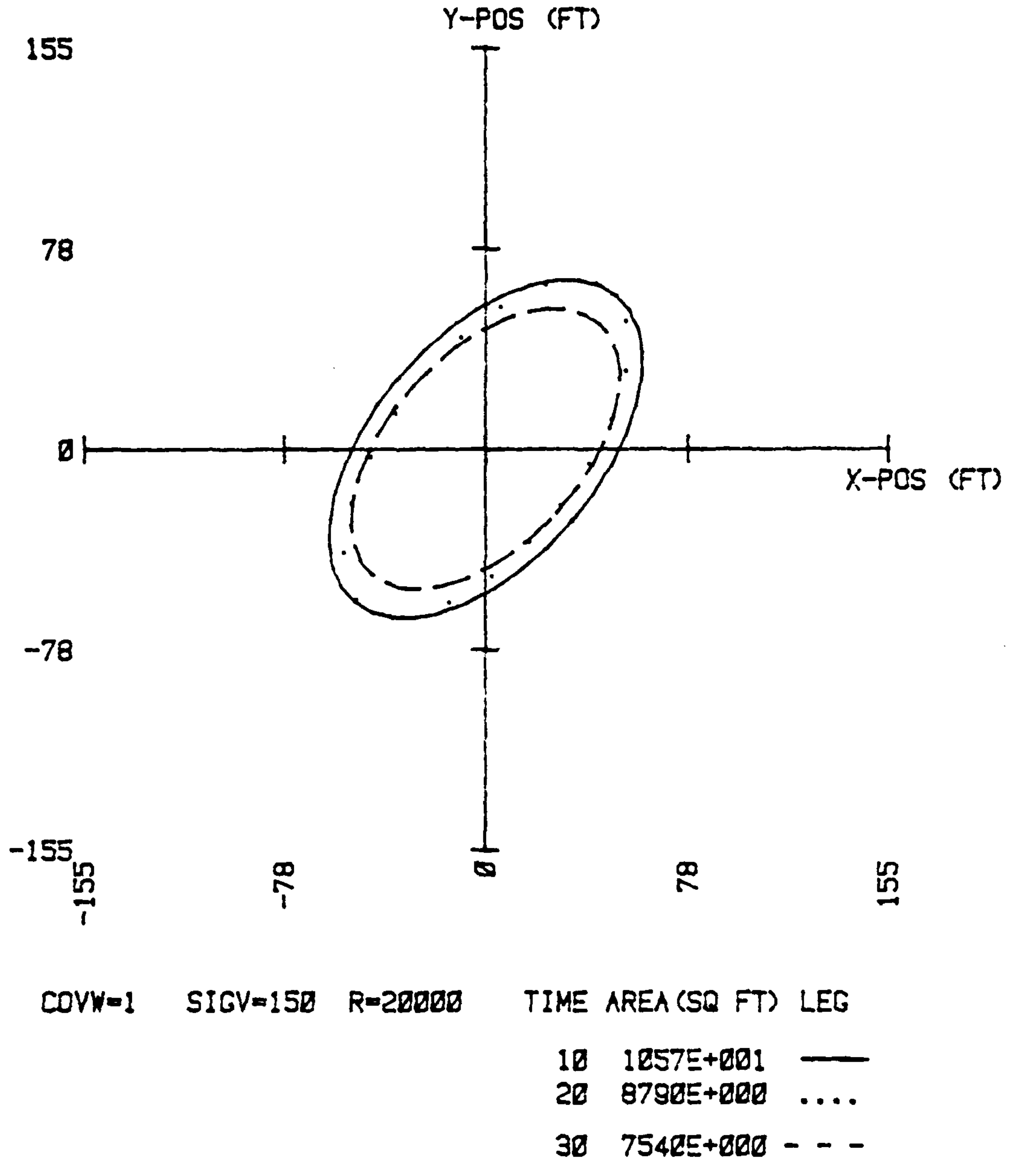


Figure 5.2 Filtered Track 1 Error Ellipses at 10 Second Increments, σ_v=150

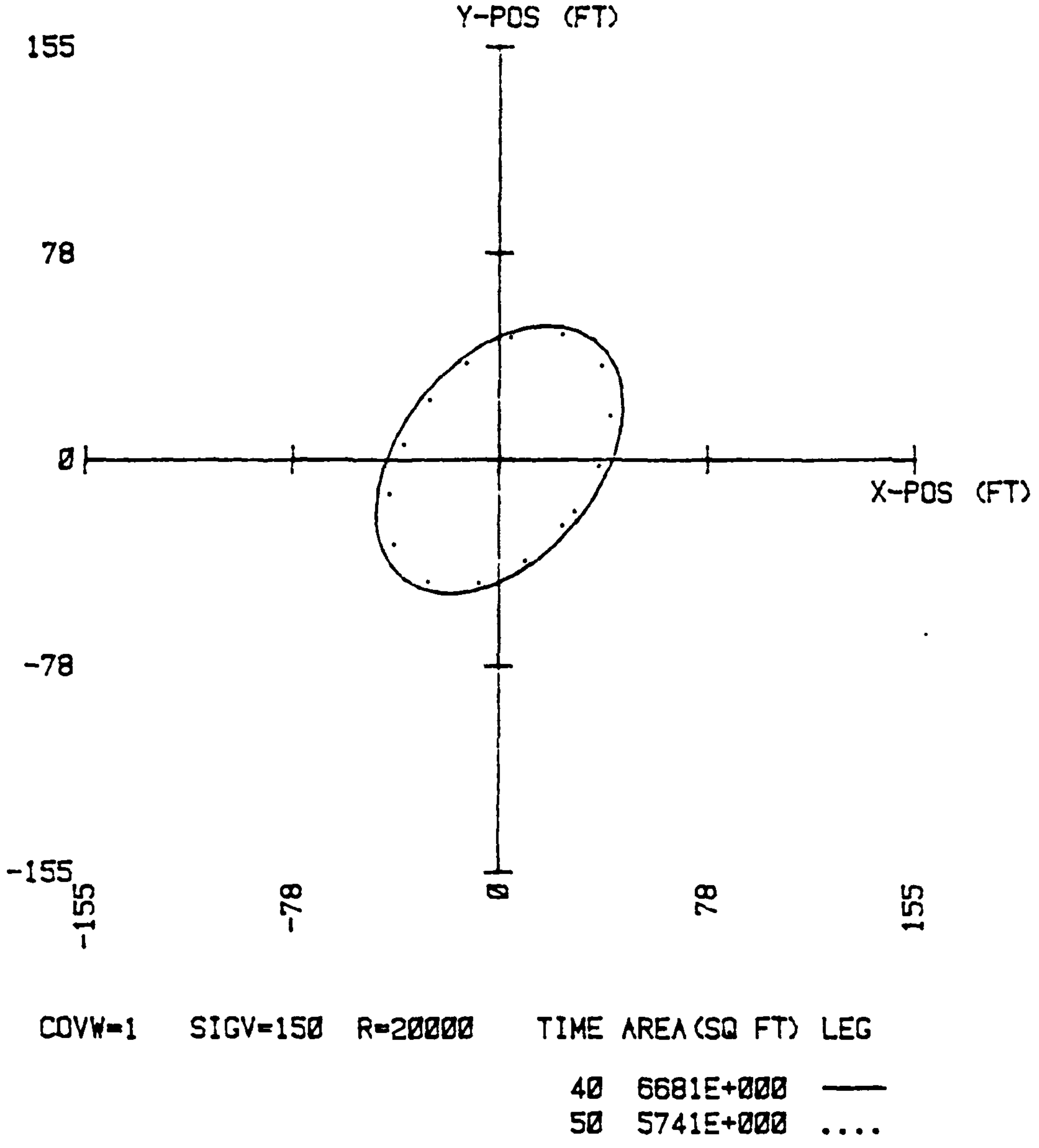


Figure 5.3 Filtered Track 1 Error Ellipses at 10 Second Increments, $\underline{\sigma}_v$=150

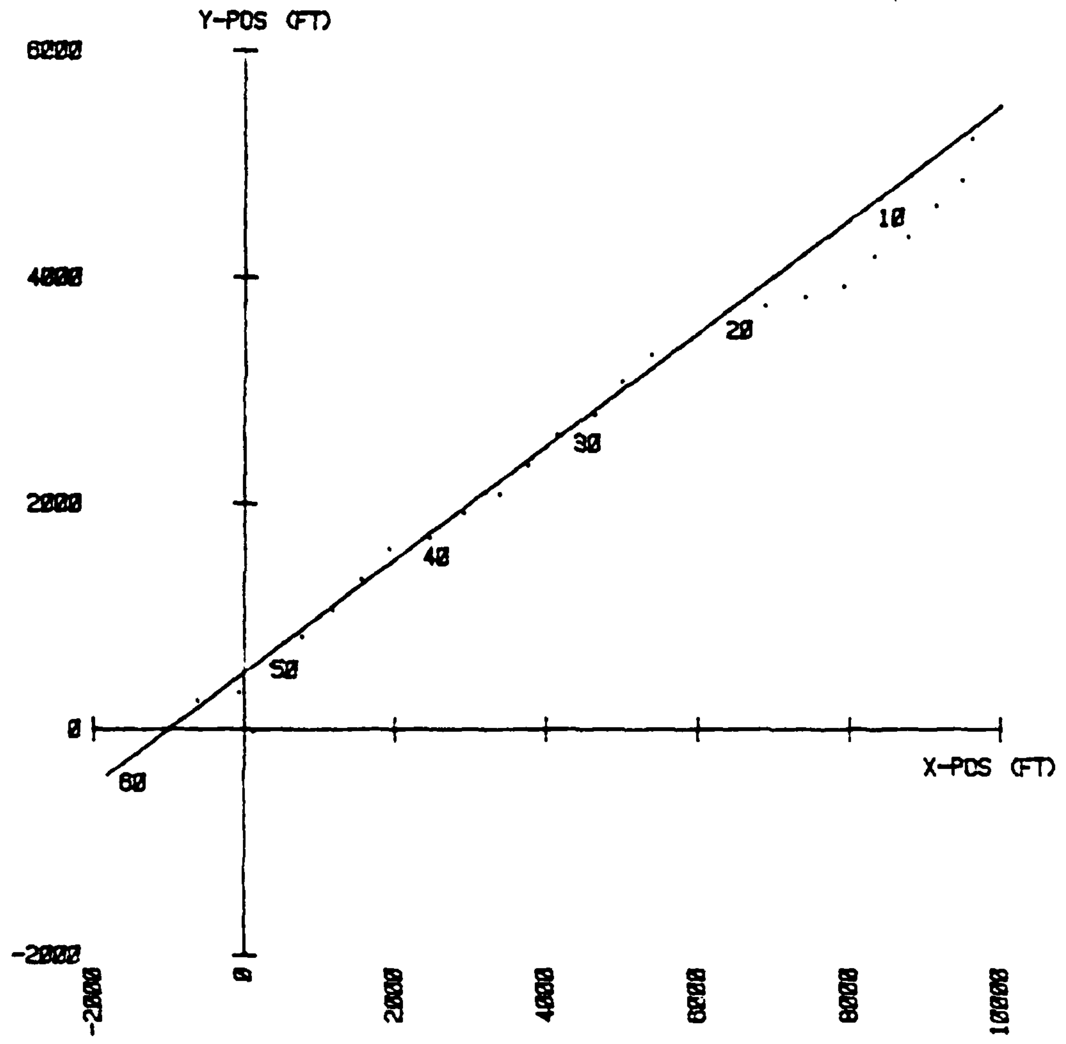
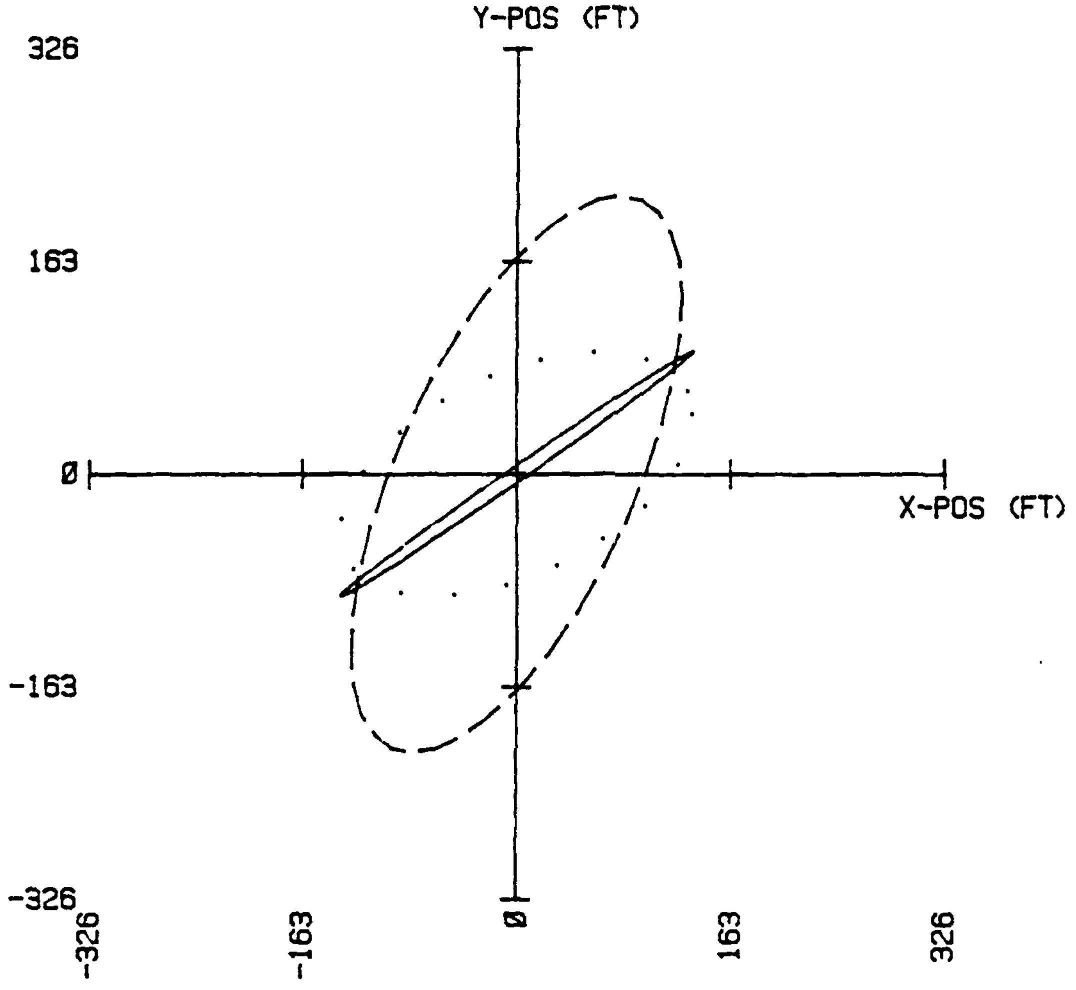


Figure 5.4 Filtered Track 1, $\underline{\sigma}_v$=300, $\underline{\sigma}_w$=1, $\underline{R}$=20,000



COVW=1Ø SIGV=3ØØ R=2ØØØØ TIME AREA(SQ FT) LEG

TIME	AREA(SQ FT)	LEG
5	2858E+ØØØ	——
15	348ØE+ØØ1	....
25	655ØE+ØØ1	- - -

Figure 5.5 Filtered Track 1 Error Ellipses for $\underline{\sigma}_v$=300

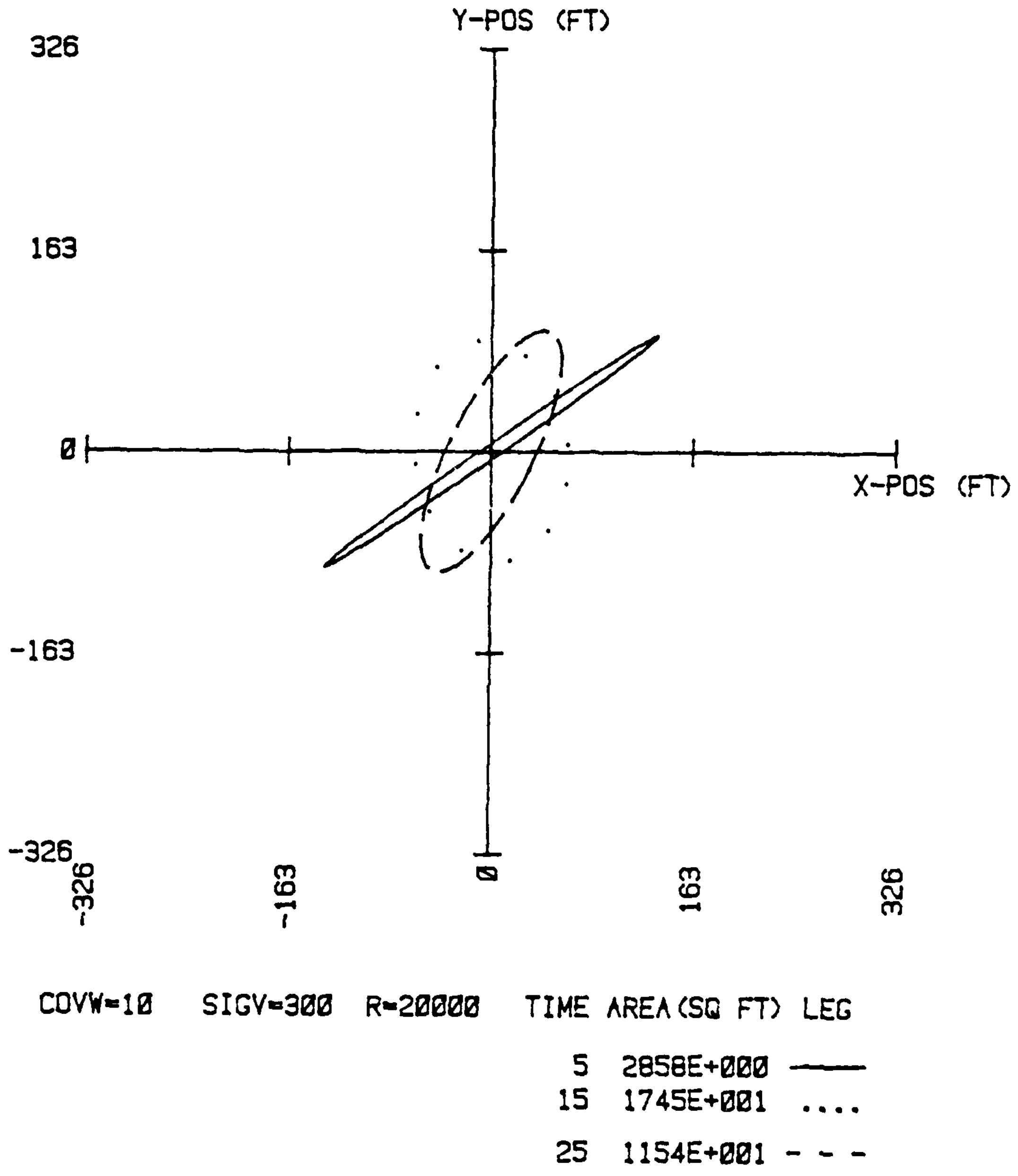


Figure 5.6 Filtered Track 1 Error Ellipses for σ_v=300, Using Statistics Window of 10

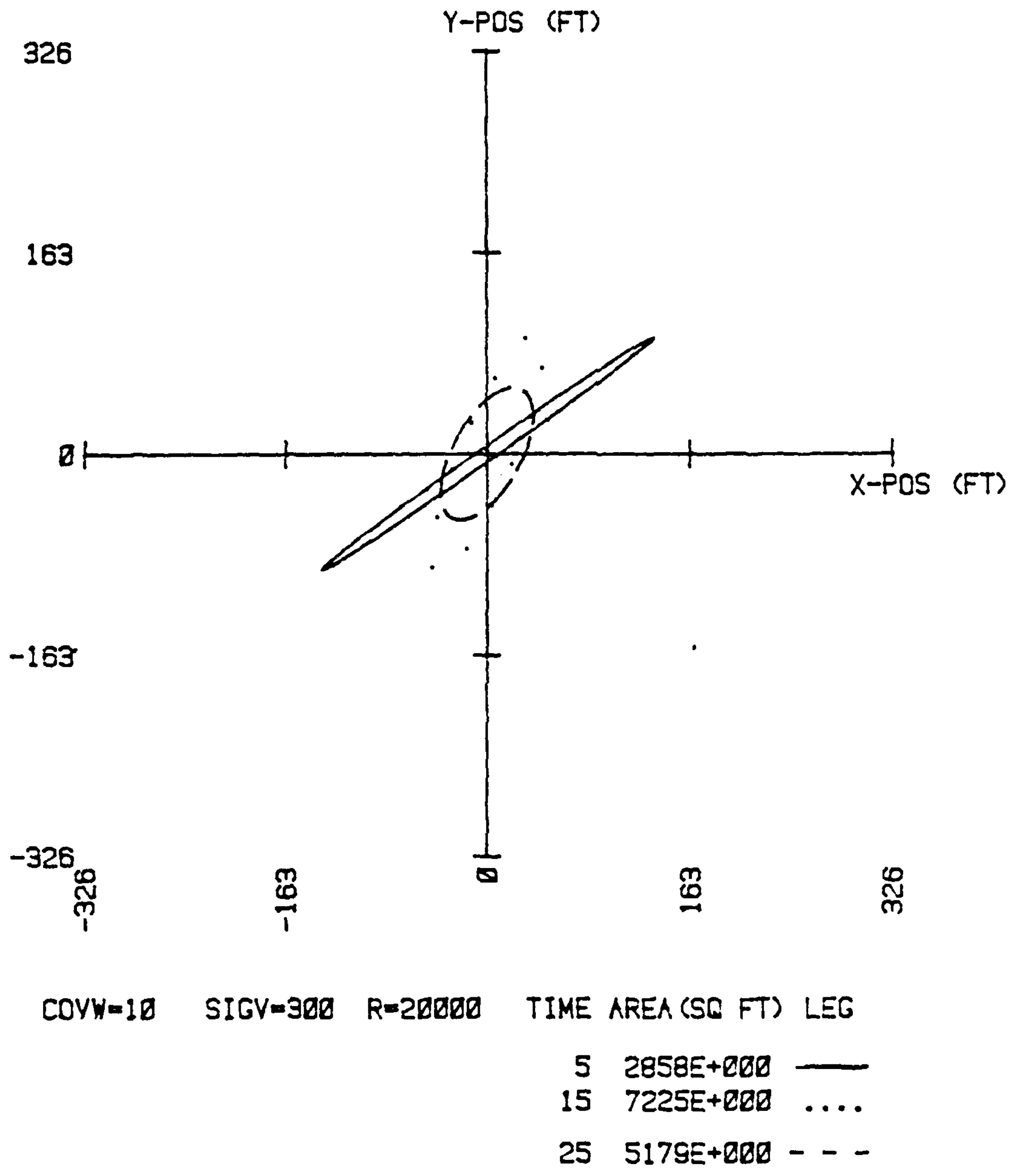


Figure 5.7 Filtered Track 1 Error Ellipses for σ_v=300, Using Statistics Window of 5

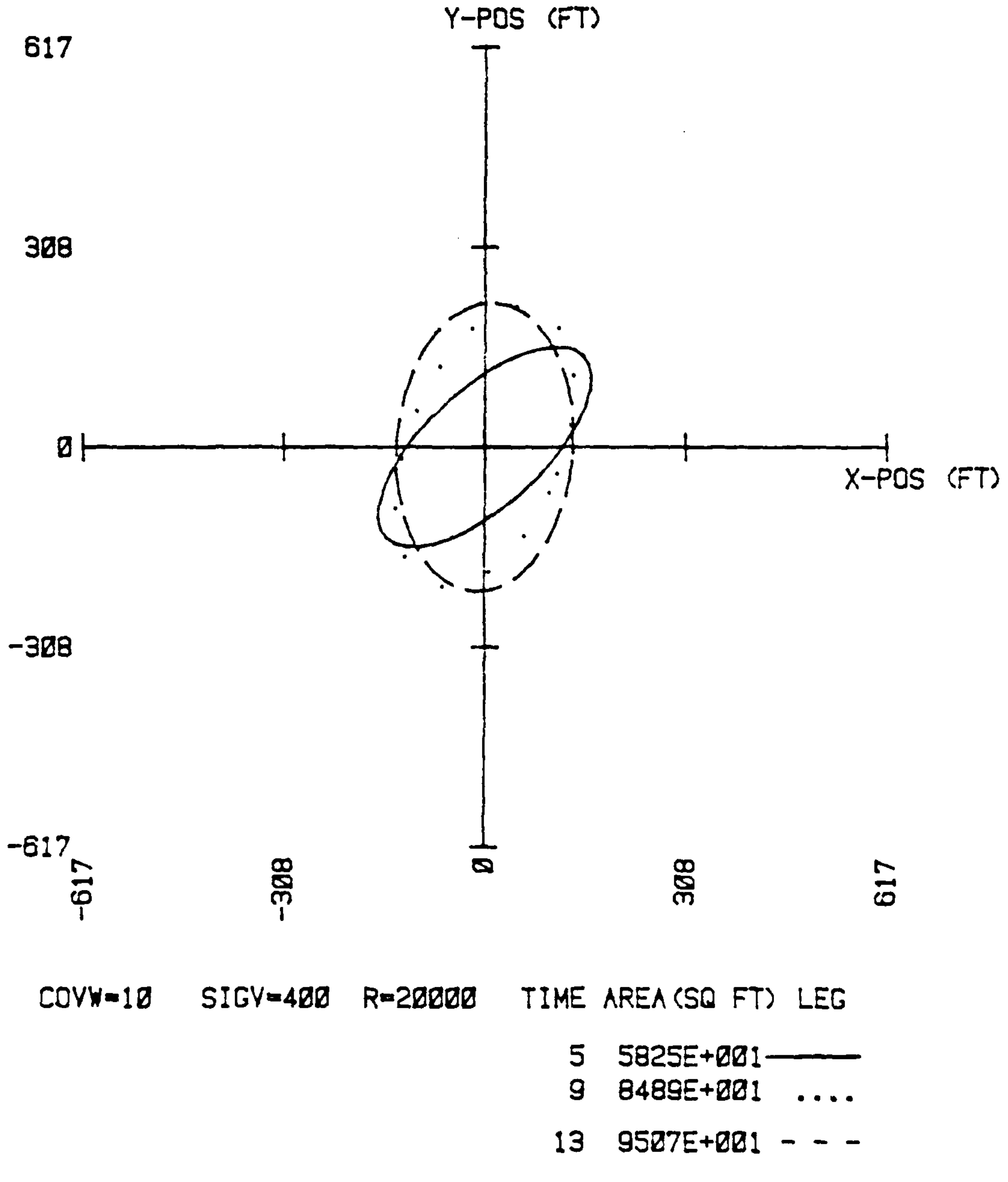


Figure 5.8 Filtered Track 1 Error Ellipse for $\underline{\sigma}_v$=400

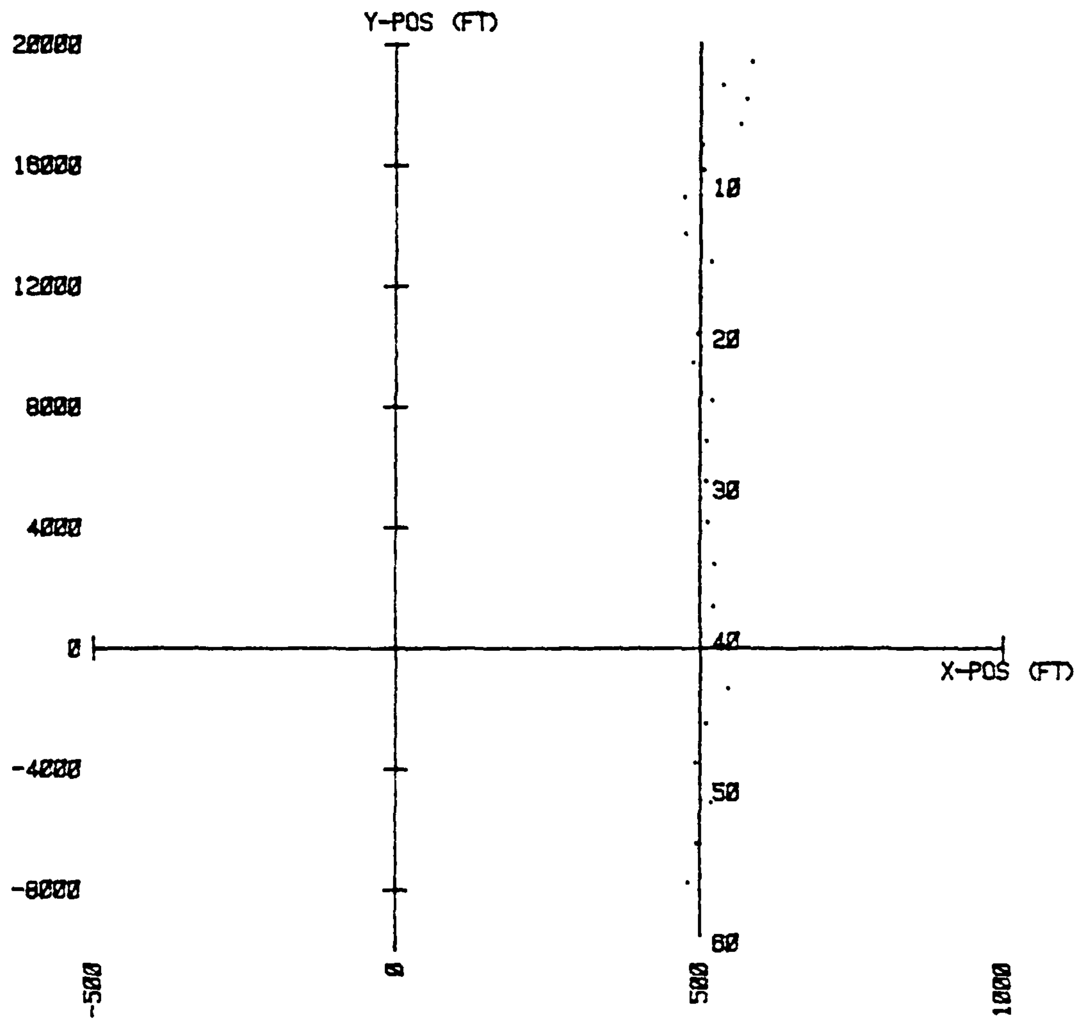


Figure 5.9 Solid Line Track 2, Vel 500 ft/sec, Dots Indicate Filtered Track for $\underline{\sigma}_v$=150, $\underline{\sigma}_w$=1, $\underline{R}$=20,000

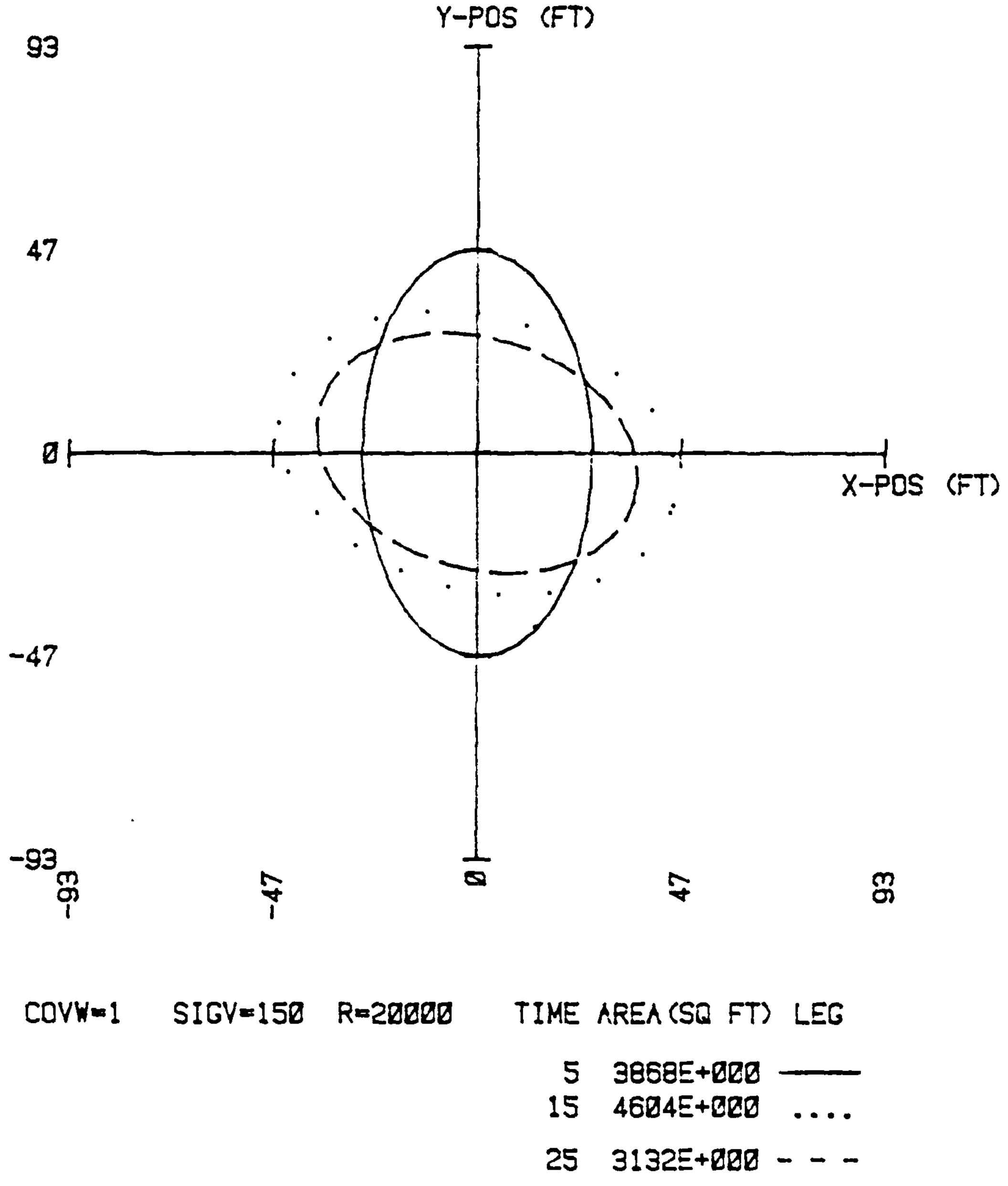
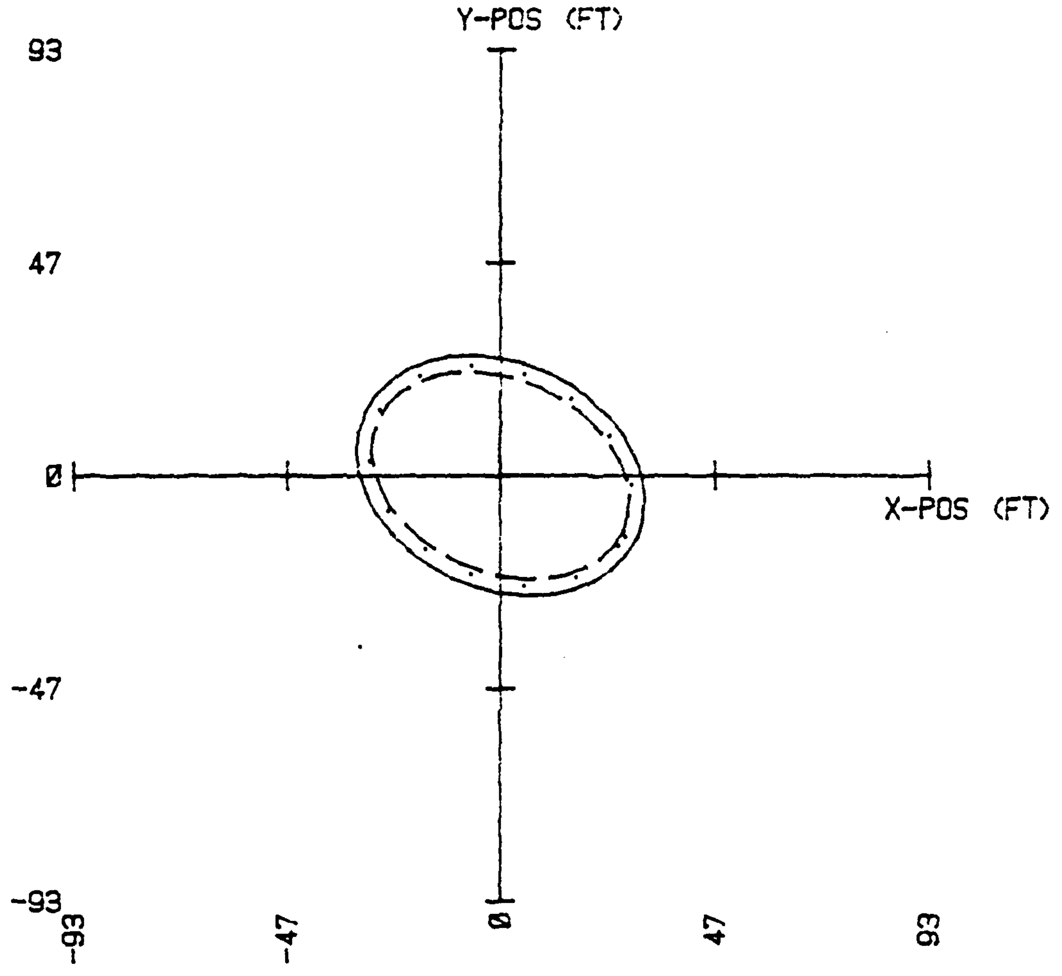


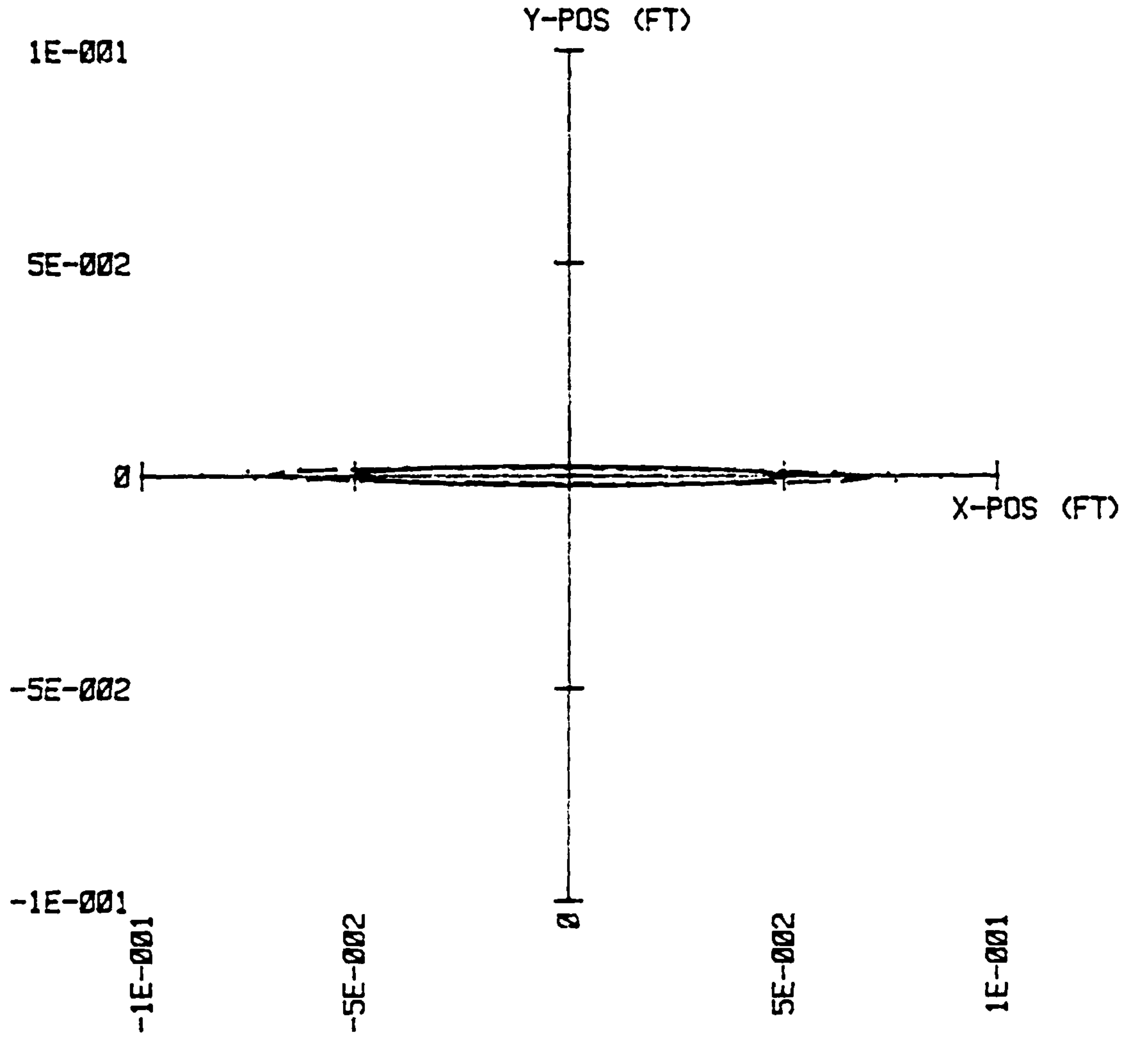
Figure 5.10 Filtered Track 2 Error Ellipses, σ_v=150



COVW=1 SIGV=150 R=20000

TIME	AREA(SQ FT)	LEG
35	2550E+000	——
45	2137E+000	....
55	1950E+000	- - -

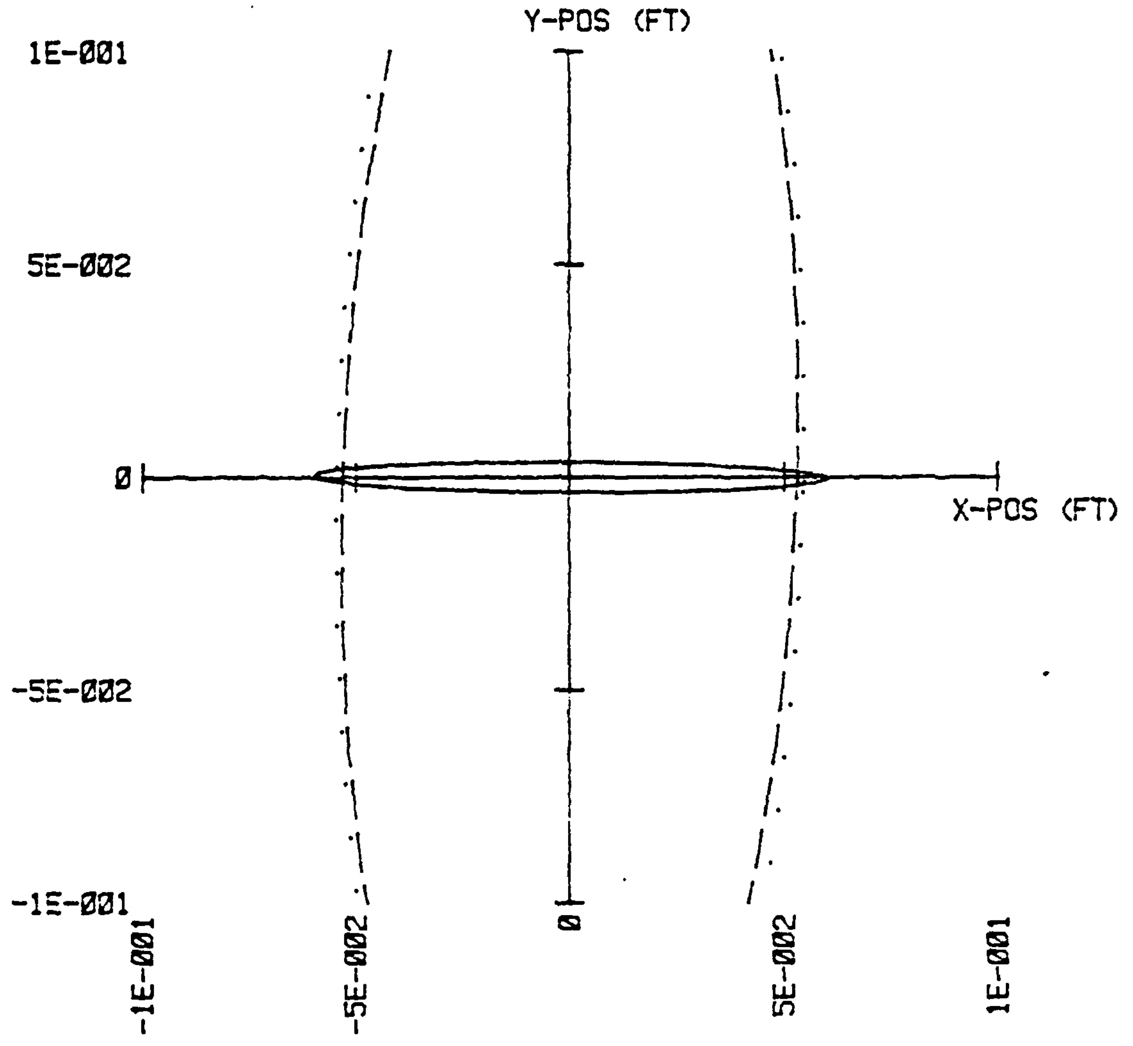
Figure 5.11 Filtered Track 2 Error Ellipses, σ_v=150



COVW=1 SIGV=150 R=20000

TIME	AREA (SQ FT)	LEG
5	4028E-007	———
15	5156E-007	....
25	4521E-007	- - -

Figure 5.12 Filtered Track 2 Error Ellipses, σ_v=150, Error Normalized



```
COVW=1   SIGV=150  R=20000   TIME AREA(SQ FT) LEG

                              35  7400E-007  ———
                              45  3869E-005  ....
                              55  3416E-005  - - -
```

Figure 5.13 Filtered Track 2 Error Ellipses, $\underline{\sigma}_v$=150, Error Normalized

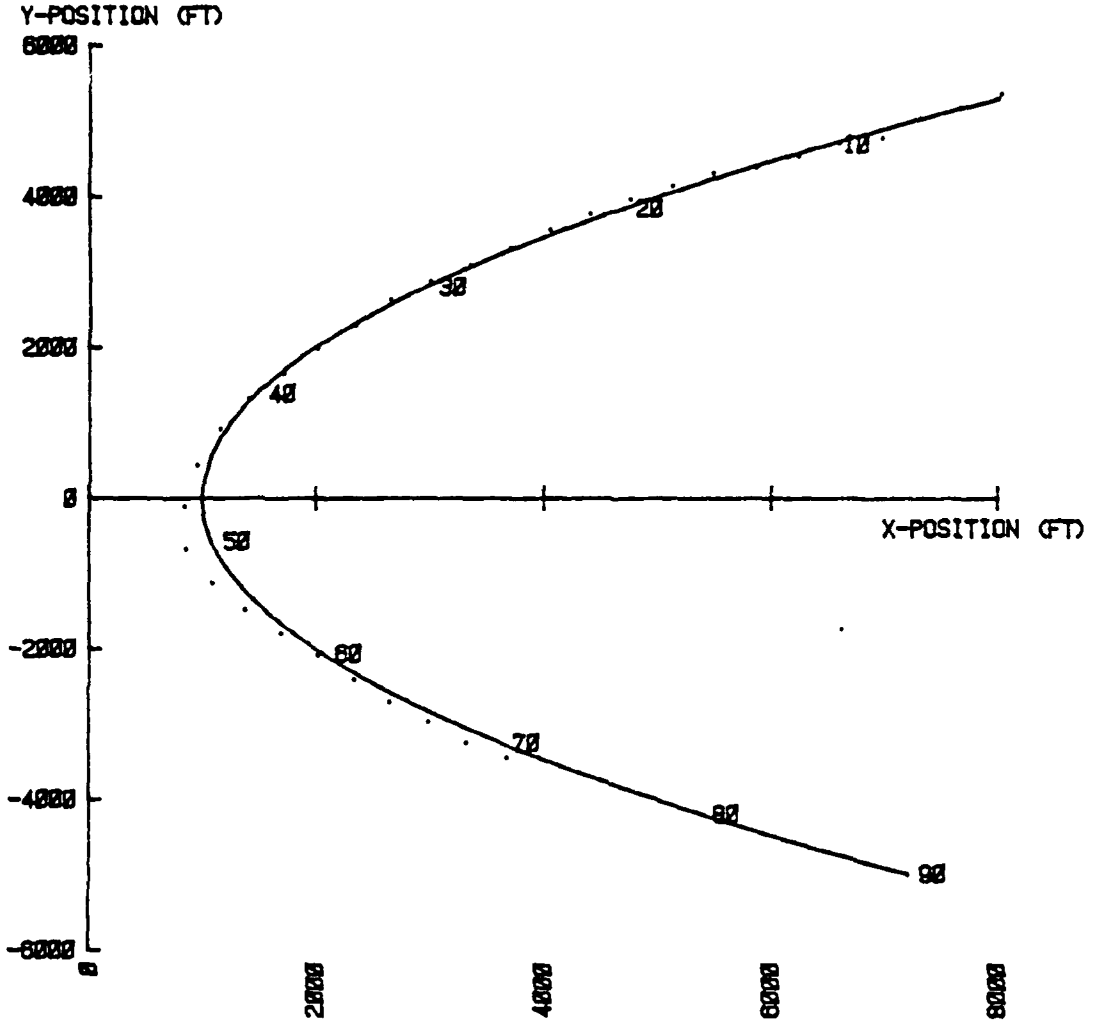


Figure 5.14 Solid Line Track 3, Vel 200 ft/sec, Dots Indicate Filtered Track for $\underline{\sigma}_v$=150, $\underline{\sigma}_w$=10, $\underline{R}$=20,000

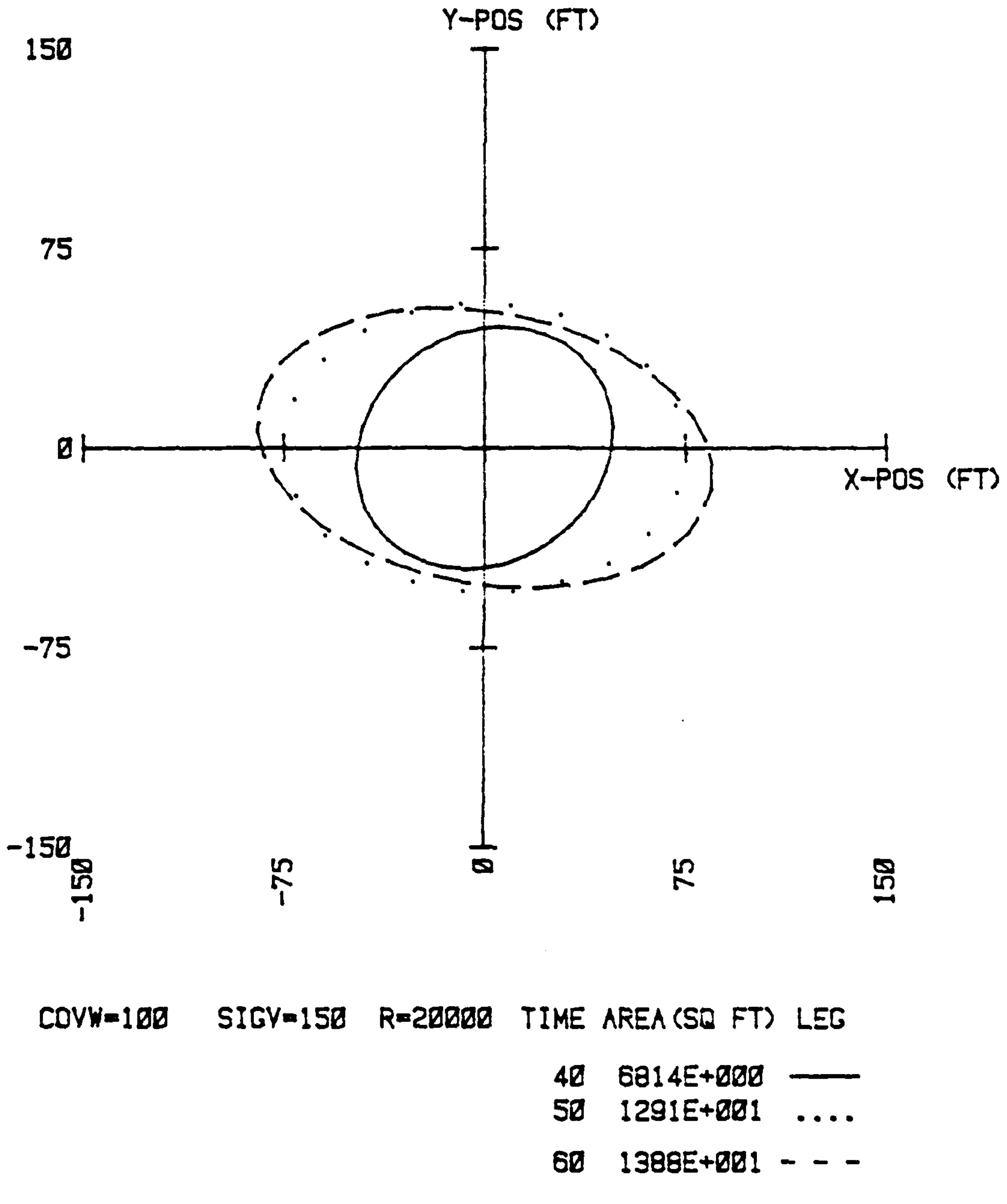


Figure 5.15 Filtered Track 3 Error Ellipses, σ_v=150

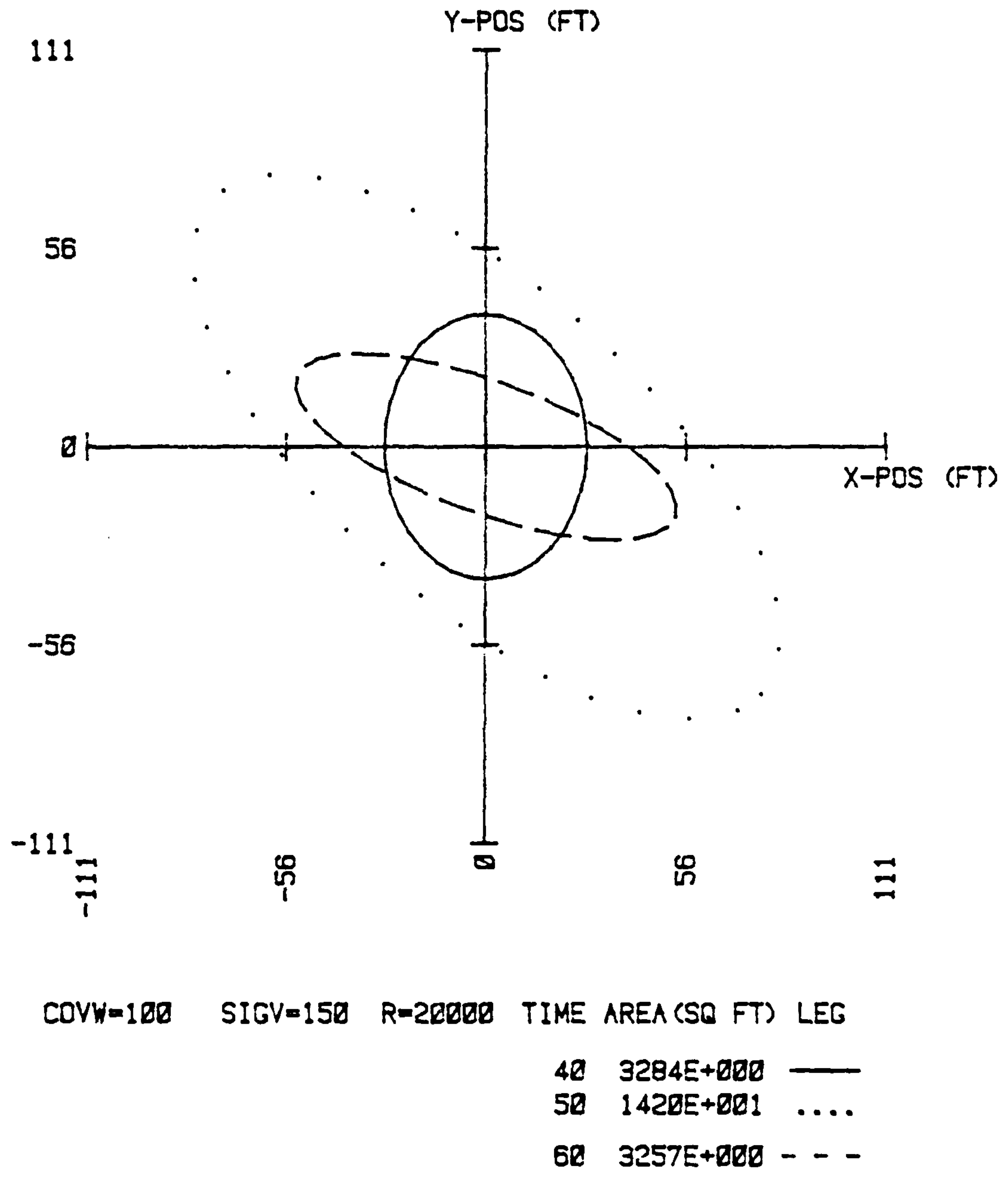


Figure 5.16 Filtered Track 3 Error Ellipses, $\underline{\sigma}_v$=150, Statistics Window 10

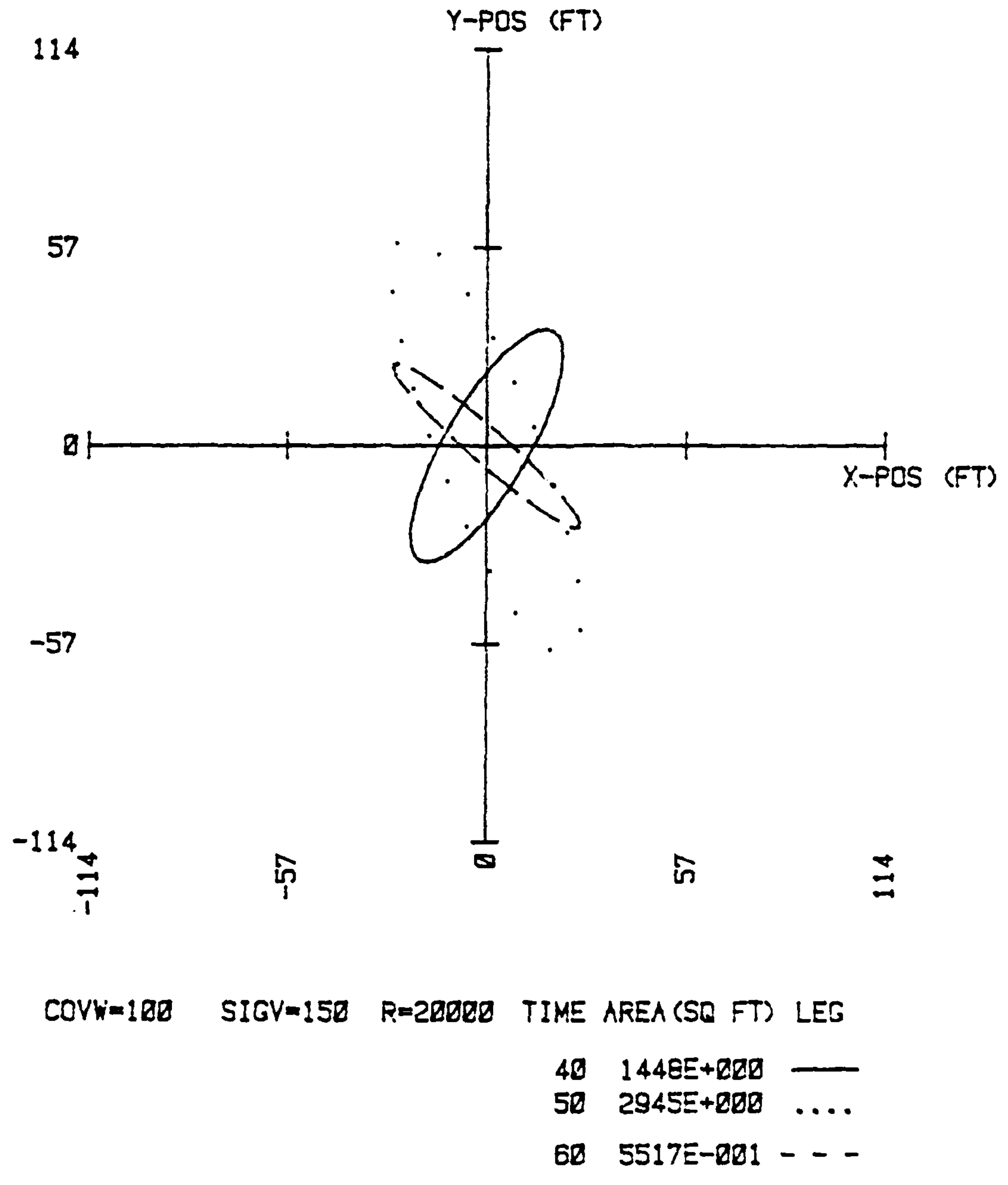


Figure 5.17 Filtered Track 3 Error Ellipses, $\underline{\sigma}_v$=150, Statistics Window 5

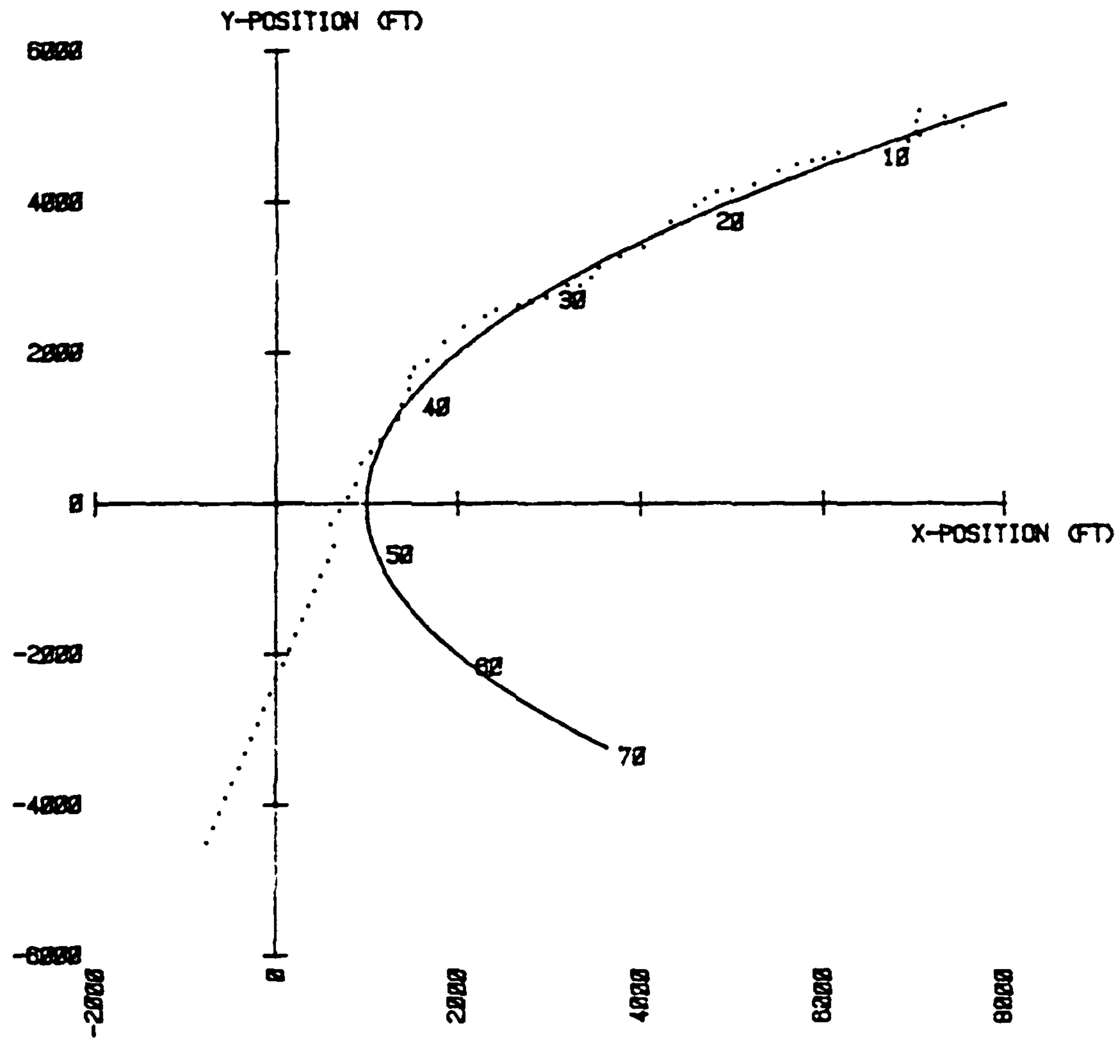


Figure 5.18 Filtered Track 3 for $\underline{\sigma}_v$=300, $\underline{\sigma}_w$=10, $\underline{R}$=20,000

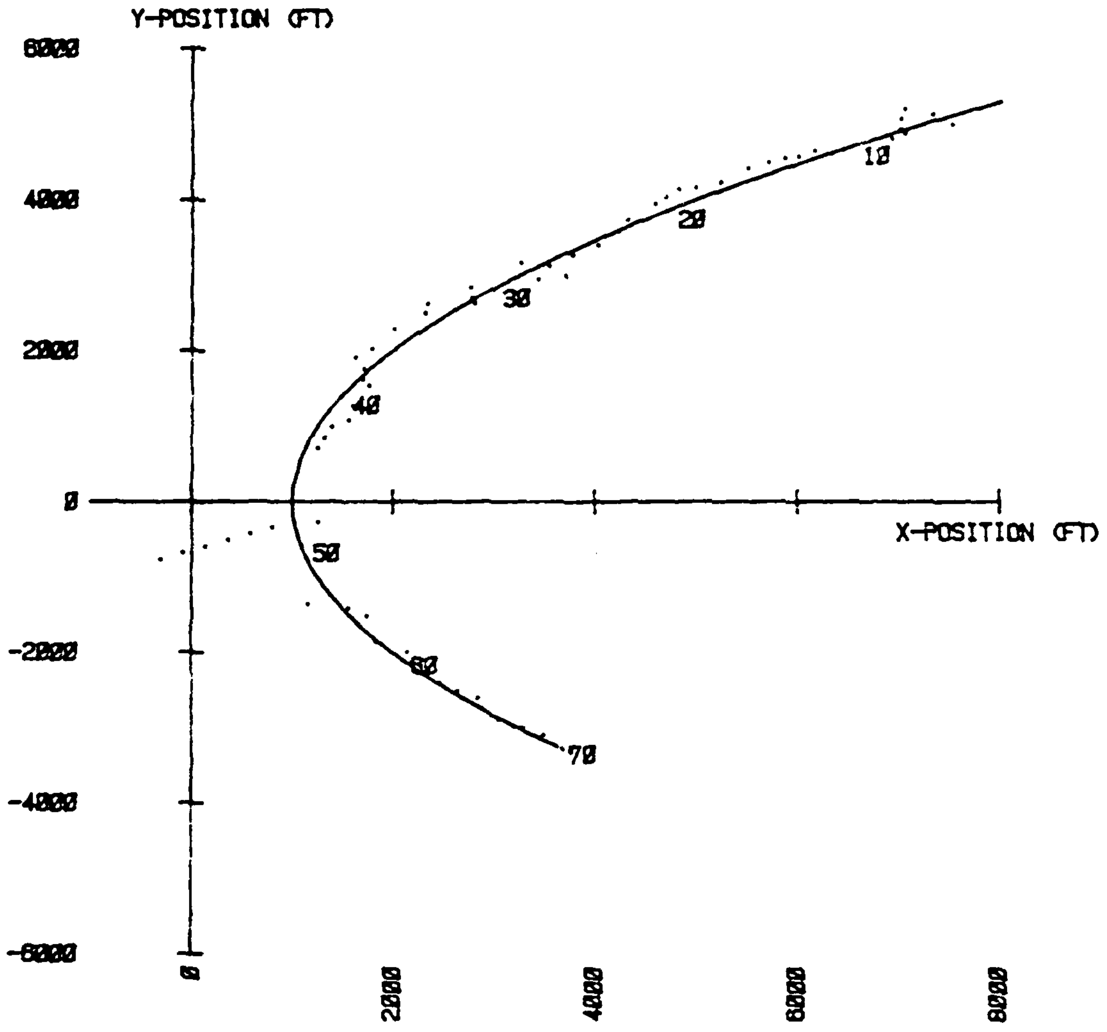
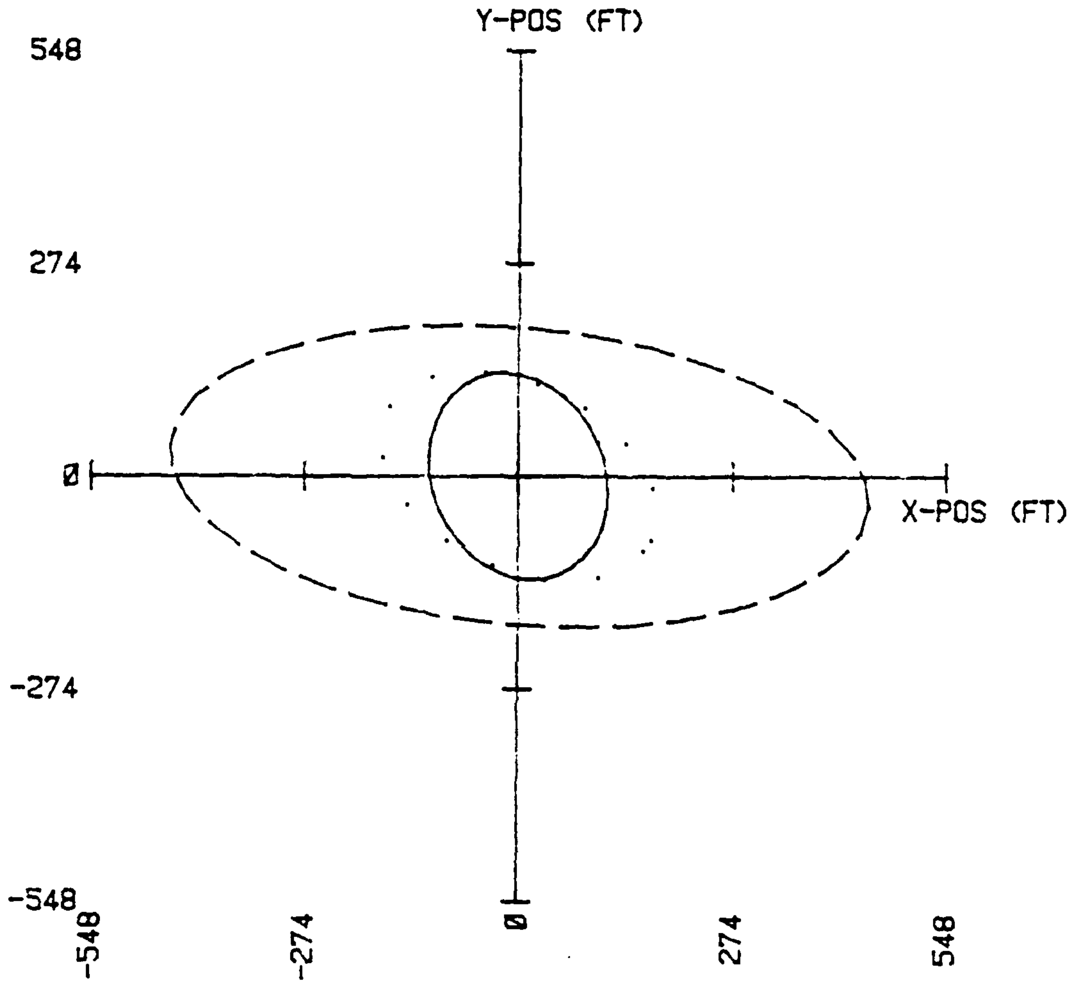
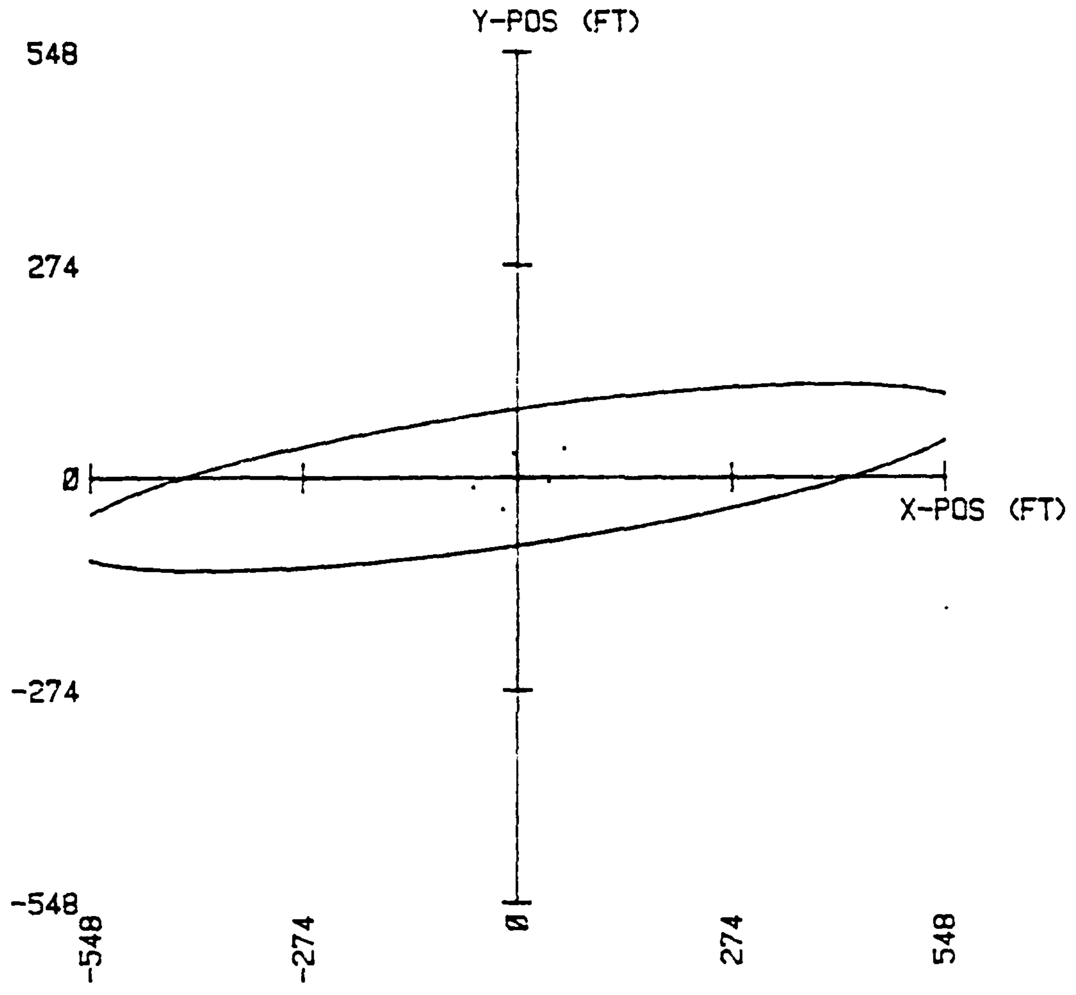


Figure 5.19 Filtered Track 3 for $\underset{\sim}{\sigma}_v$=300, $\underset{\sim}{\sigma}_w$=10, $\underset{\sim}{R}$=20,000, Statistics Window 10, Reinitialized at 5 Consecutive Ellipse Area Increments



```
COVW=1ØØ   SIGV=3ØØ  R=2ØØØØ  TIME AREA(SQ FT) LEG RESET
                              3Ø   4734E+ØØ1 ——— 29
                              4Ø   7Ø35E+ØØ1 .... 35
                              5Ø   2698E+ØØ2 - - - 44
```

Figure 5.20 Filtered Track 3 Error Ellipses, σ_v=300, Statistics Window 10, Reinitialize 5



```
COVW=1ØØ   SIGV=3ØØ  R=2ØØØØ  TIME  AREA(SQ FT)  LEG RESET

                               6Ø   1617E+ØØ2   ---- 52
                               7Ø   6563E+ØØØ   .... 52
```

Figure 5.21 Filtered Track 3 Error Ellipses, σ_v=300, Statistics Window 10, Reinitialize 5

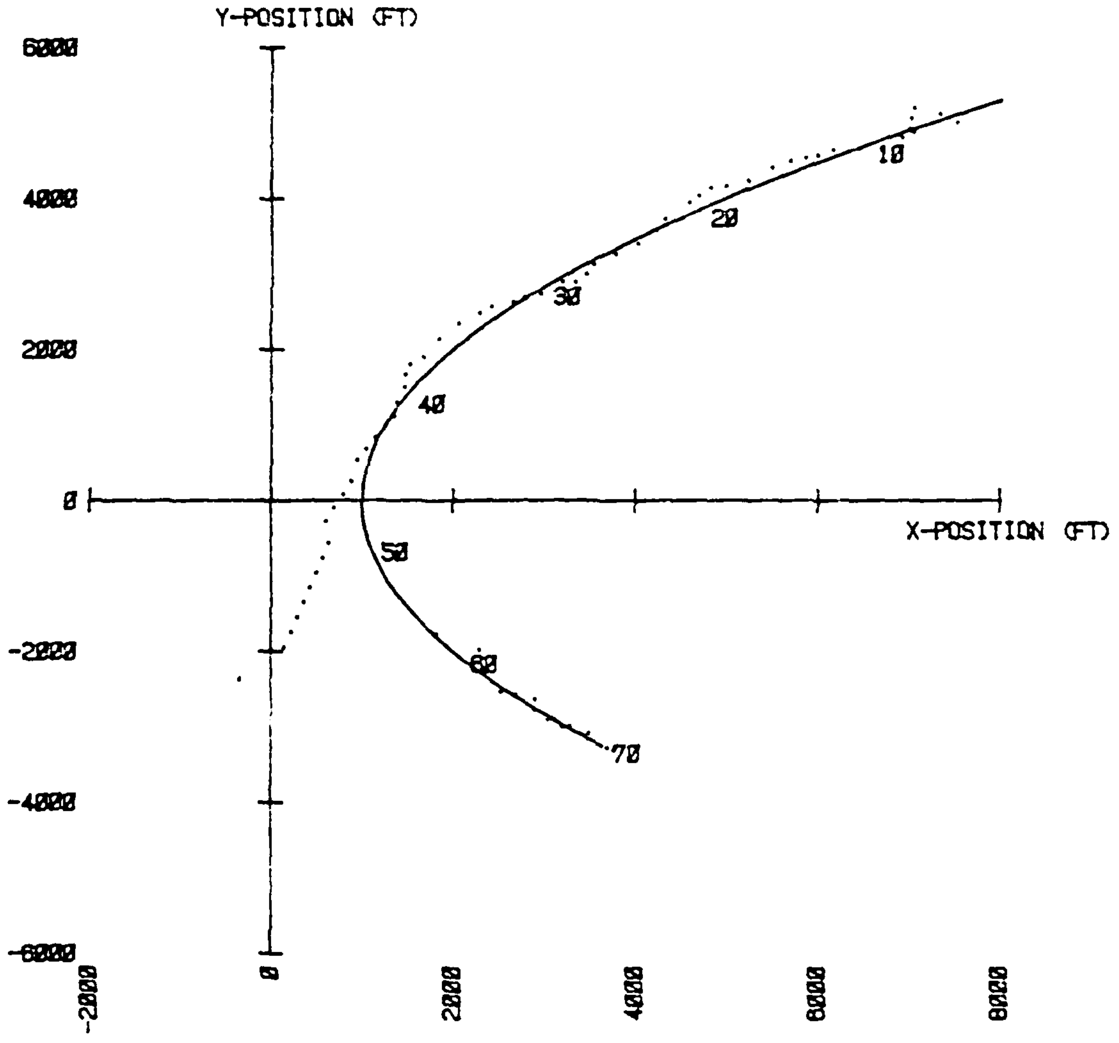
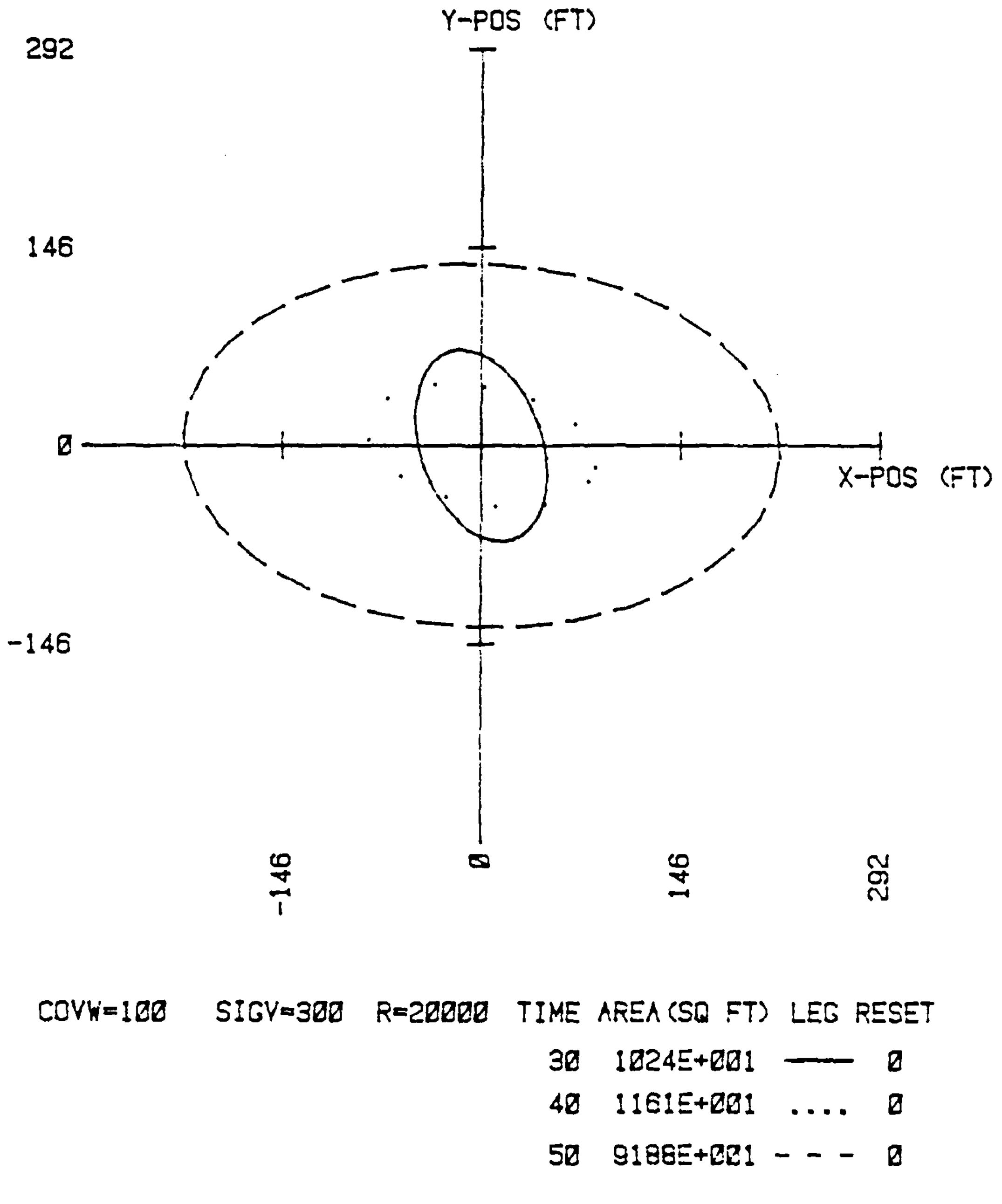
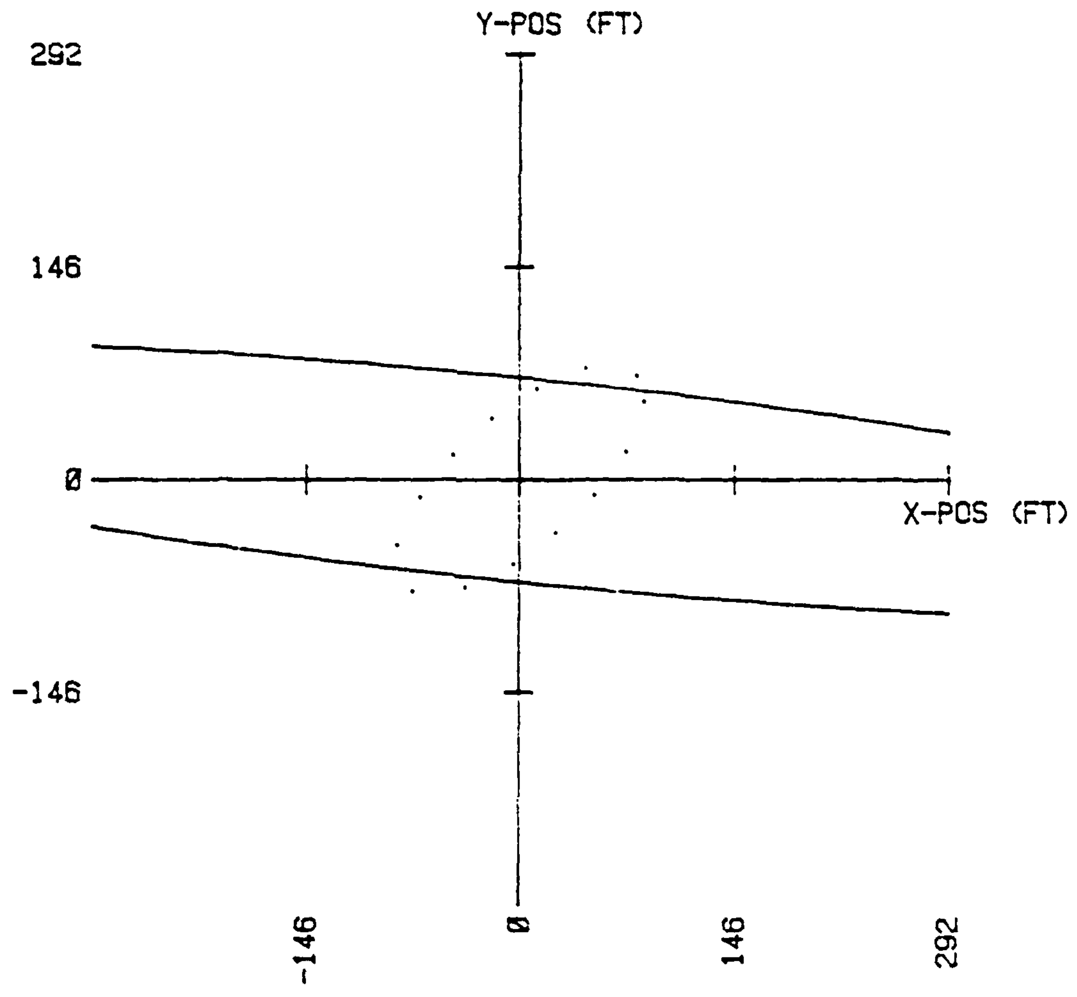


Figure 5.22 Filtered Track 3 for $\underline{\sigma}_v$=300, $\underline{\sigma}_w$=10, $\underline{R}$=20,000
Statistics Window 10, Reinitialize 10



igure 5.23 Filtered Track 3 Error Ellipses, $\underline{\sigma}_v$=300, Statistics Window 10, Reinitialize 10



```
COVW=100   SIGV=300  R=20000  TIME  AREA(SQ FT)  LEG RESET
                                60  1380E+002  ——— 57
                                70  1478E+001  .... 57
```

Figure 5.24 Filtered Track 3 Error Ellipses, $\underline{\sigma}_v$=300, Statistics Window 10, Reinitialize 10

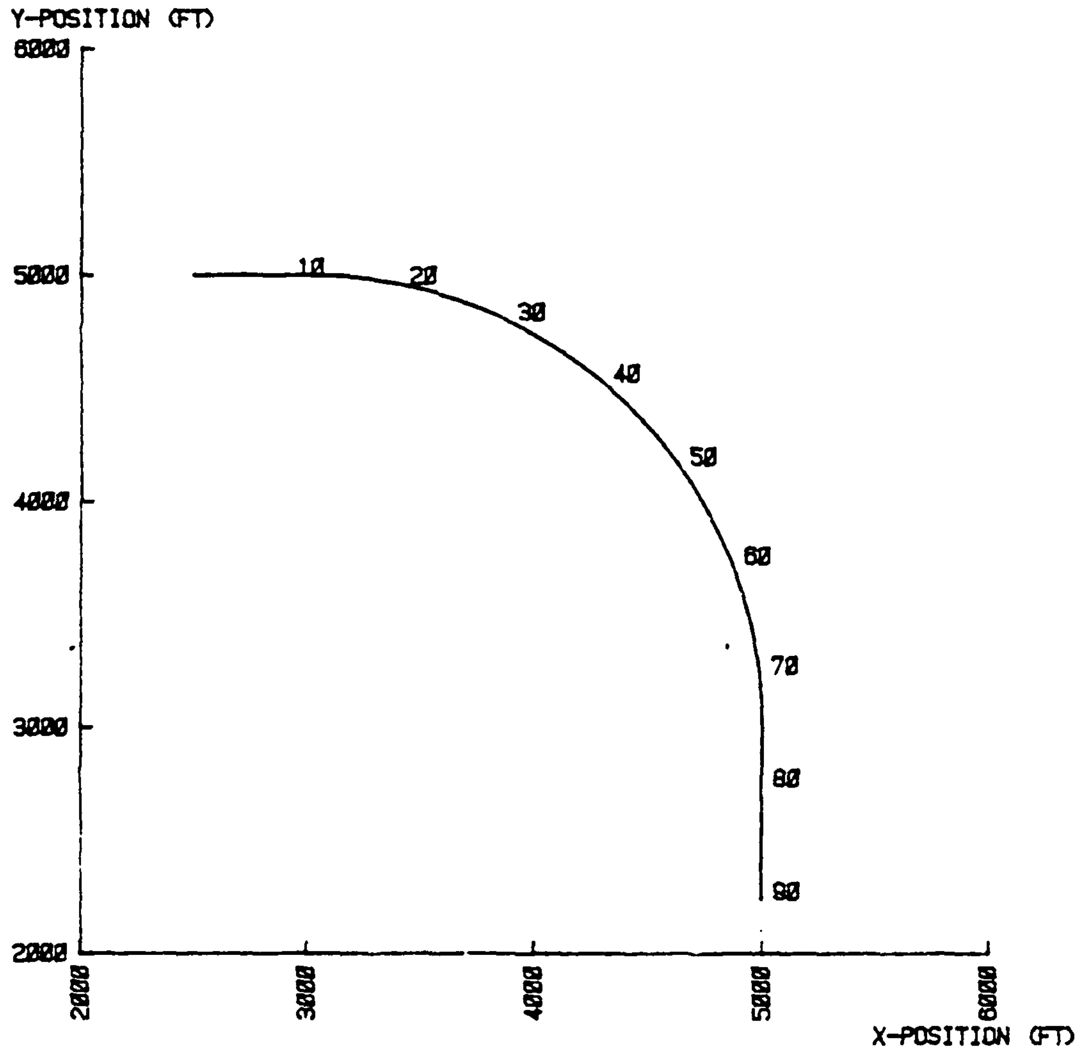


Figure 5.25 Track 4, Vel 50 ft/sec

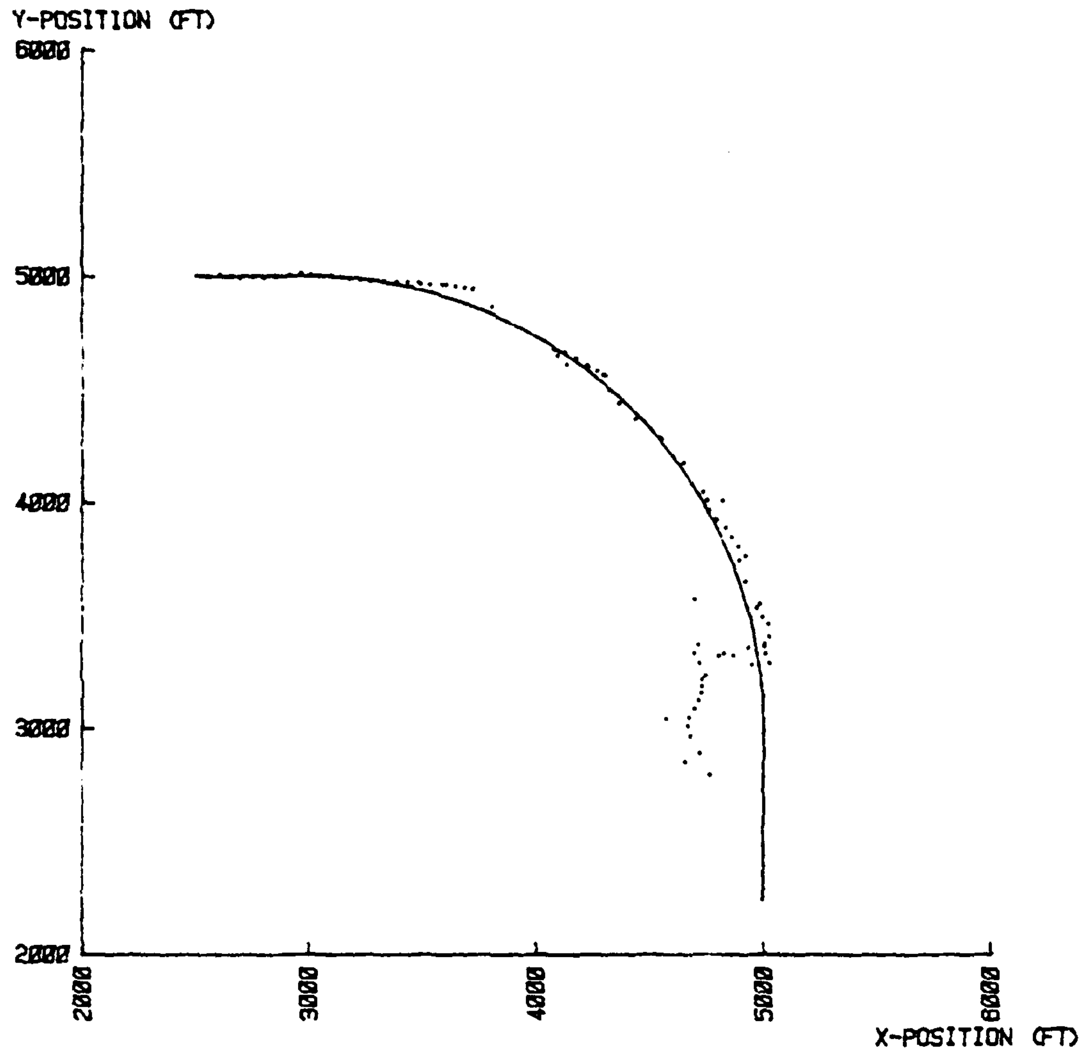


Figure 5.26 Filtered Track 4, $\underline{\sigma}_v$=30, $\underline{R}$=900, $\underline{\sigma}_w^2$=150

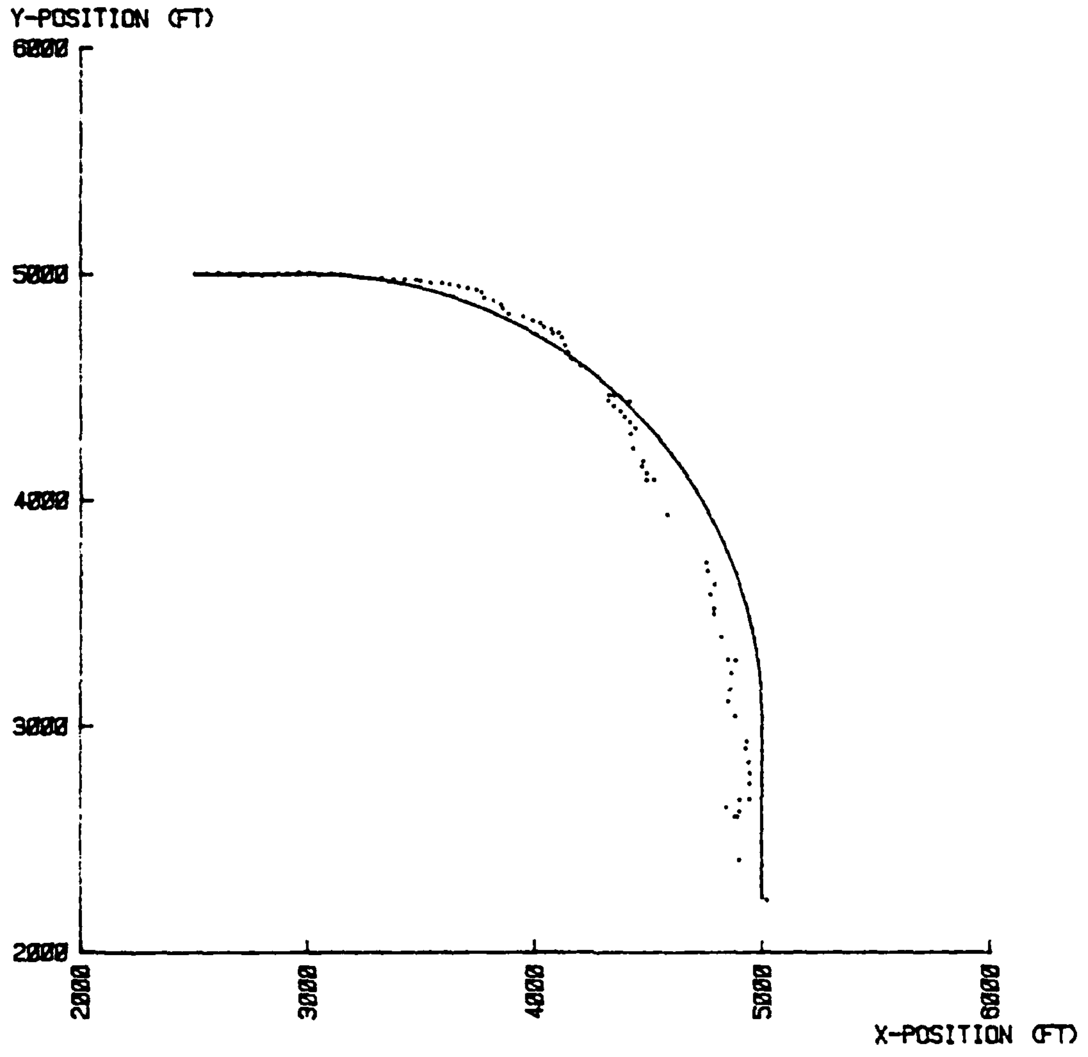
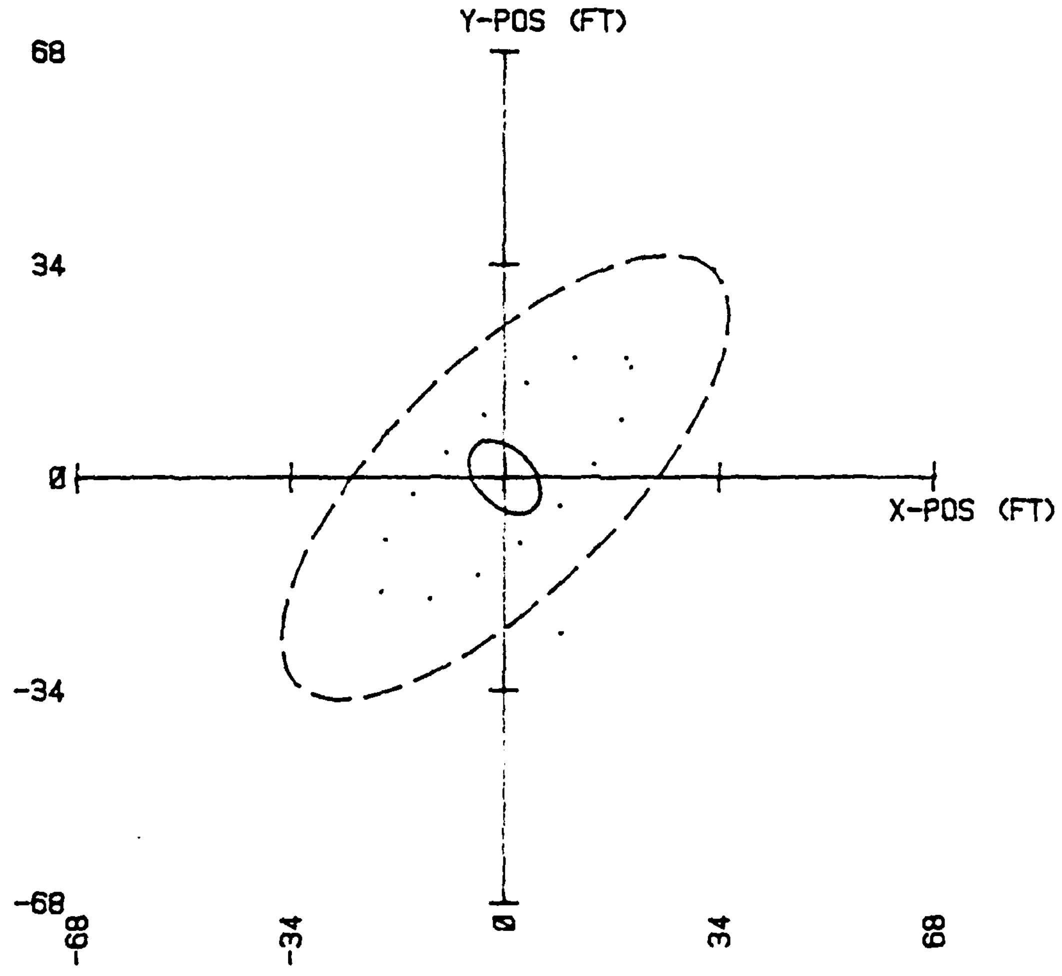


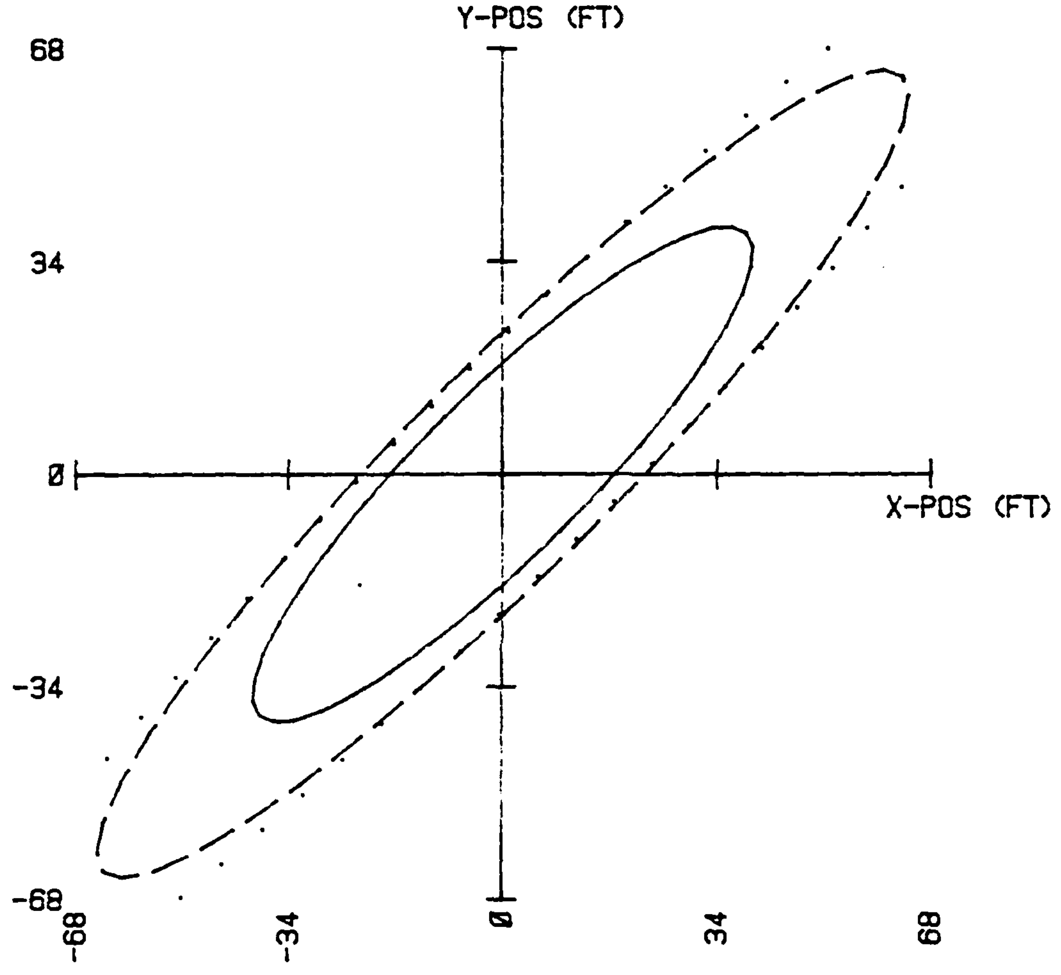
Figure 5.27 Filtered Track 4, $\underline{\sigma}_v$=30, $\underline{R}$=900, Statistics Window 10, Varying $\underline{\sigma}_w^2$



SIGV = 30 R = 900

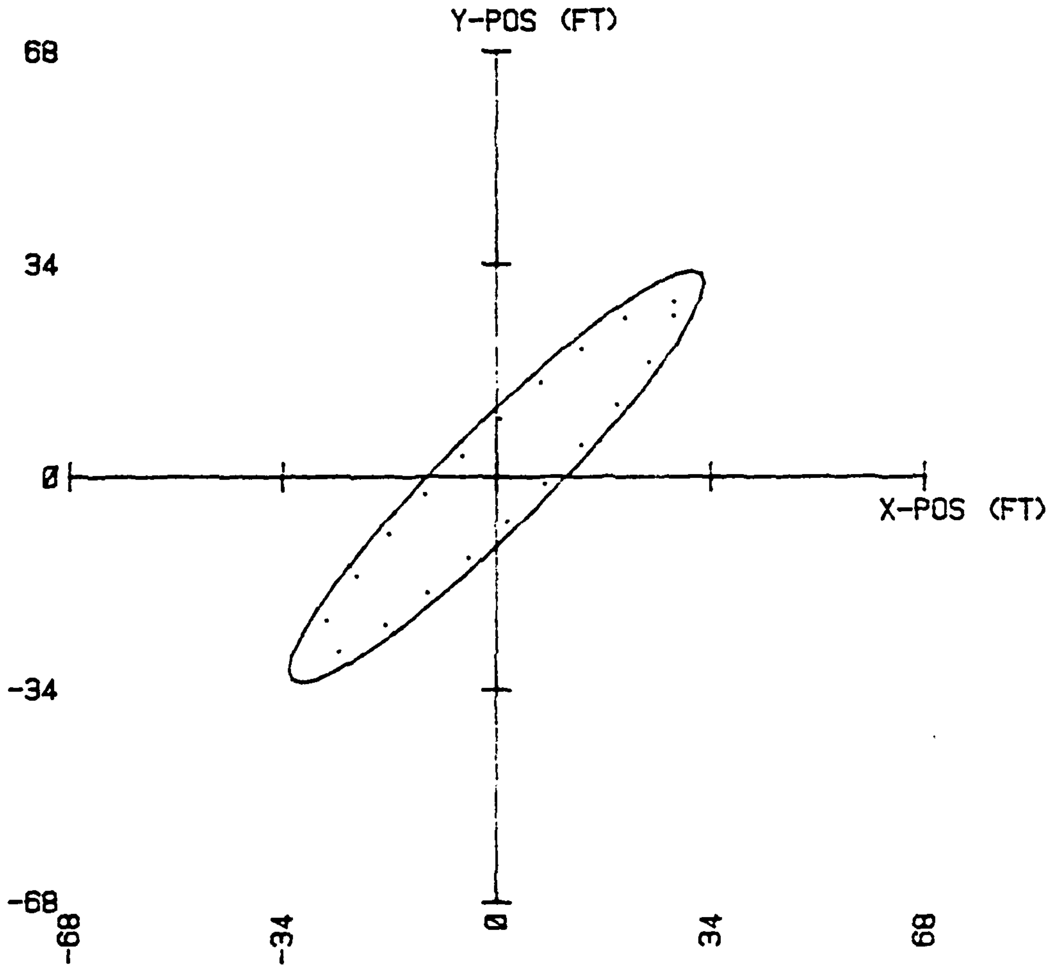
TIME	AREA (SQ FT)	LEG.	COVW
15	9688E-002	——	20
25	8097E-001	....	40
35	2740E+000	_ _ _	160

Figure 5.28 Filtered Track 4 Error Ellipses, Statistics Window 10, Varying COVW



```
SIGV = 3Ø  R = 9ØØ    TIME  AREA(SQ FT)  LEG.  COVW
                       45   2239E+ØØØ   ———   16Ø
                       55   63Ø3E+ØØØ   ....  16Ø
                       65   4625E+ØØØ   _ _ _  8Ø
```

Figure 5.29 Filtered Track 4 Error Ellipses, Statistics Window 10, Varying COVW



```
SIGV = 30  R = 900    TIME  AREA(SQ FT)  LEG.  COVW
                       75   1163E+000    ——     40
                       85   7916E-001    ....   40
```

Figure 5.30 Filtered Track 4 Error Ellipses, Statistics Window 10, Varying COVW

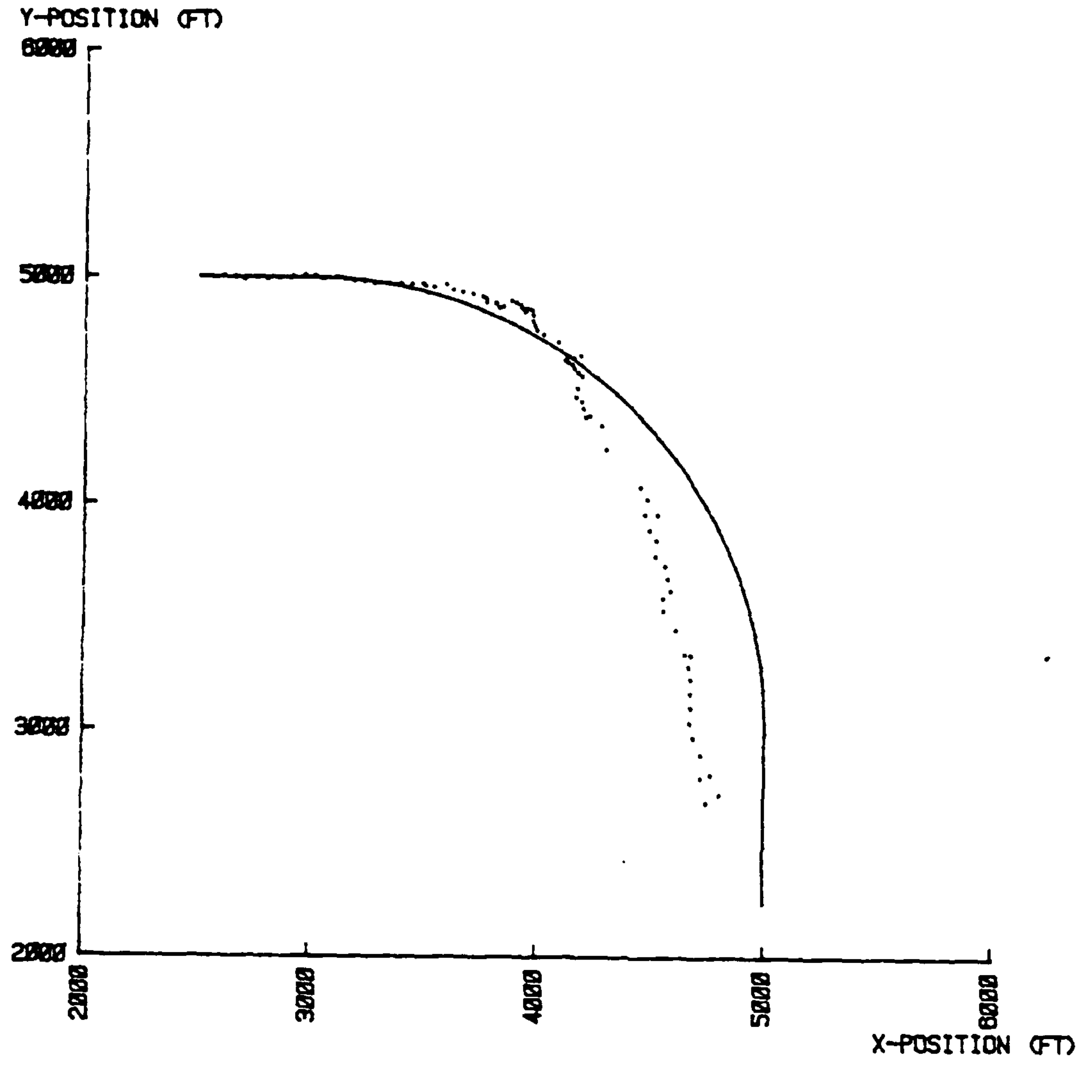
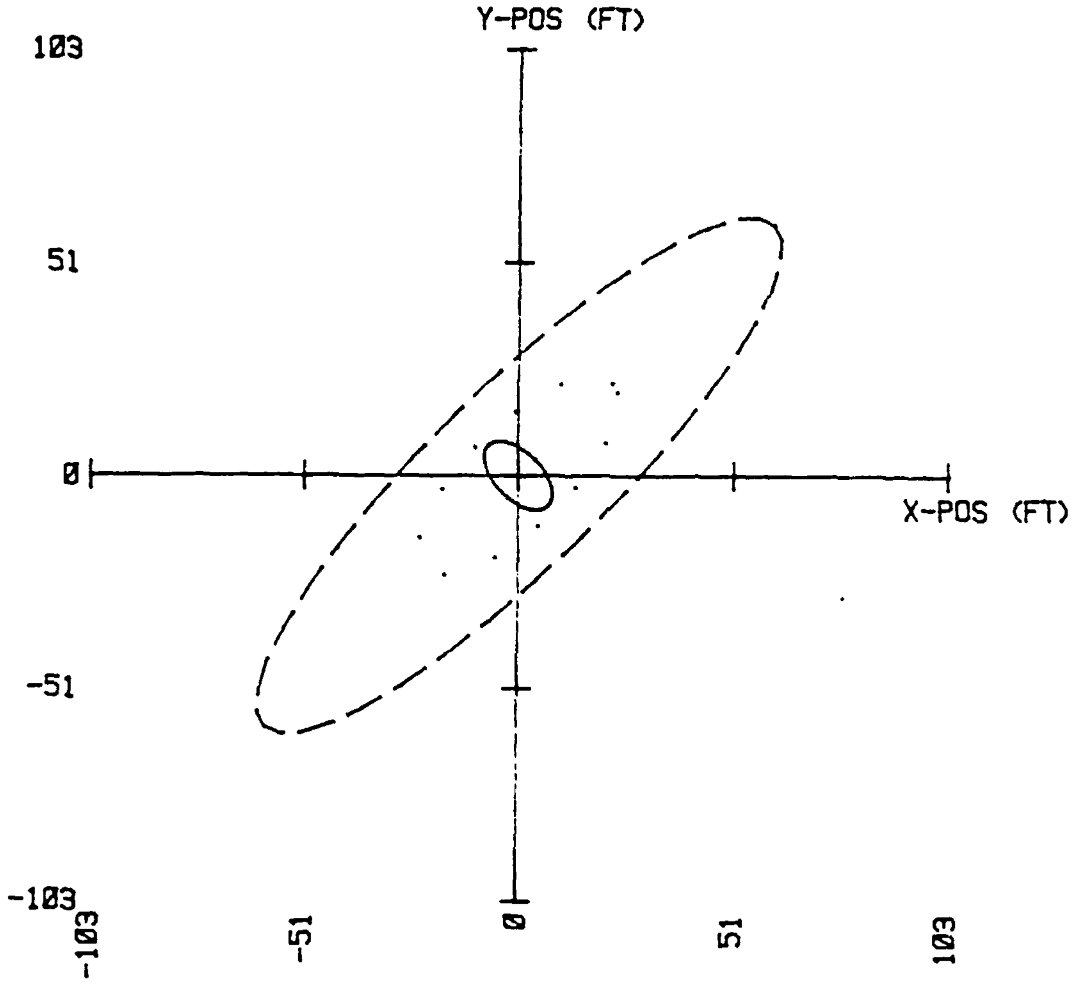


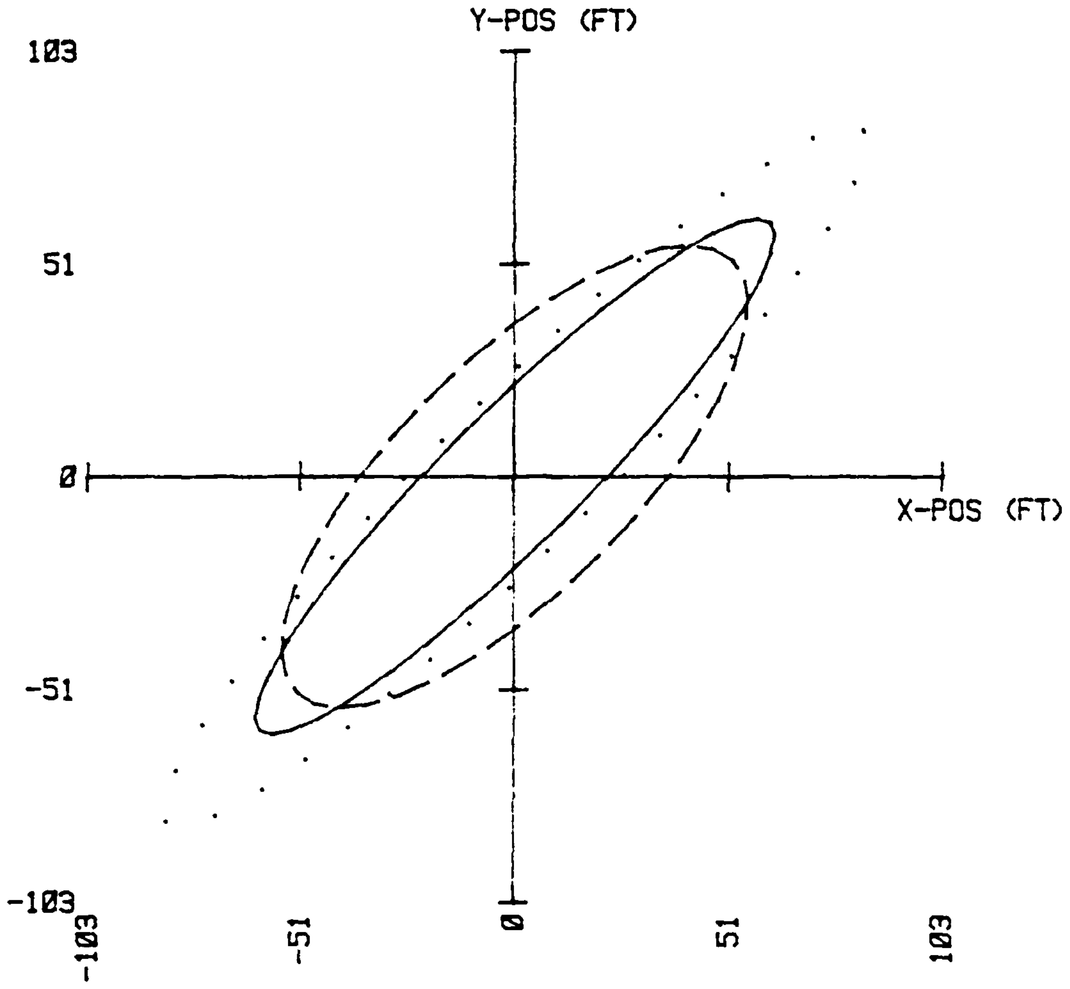
Figure 5.31 Filtered Track 4, $\underline{\sigma}_v$=30, $\underline{R}$=900, $\underline{\sigma}_w^2$=150, Reinitialize 5



SIGV=30 R=900 COVW=150

TIME	AREA (SQ FT)	LEG.	RESET
15	1805E-001	——	0
25	1211E+000	....	23
35	5743E+000	_ _ _	33

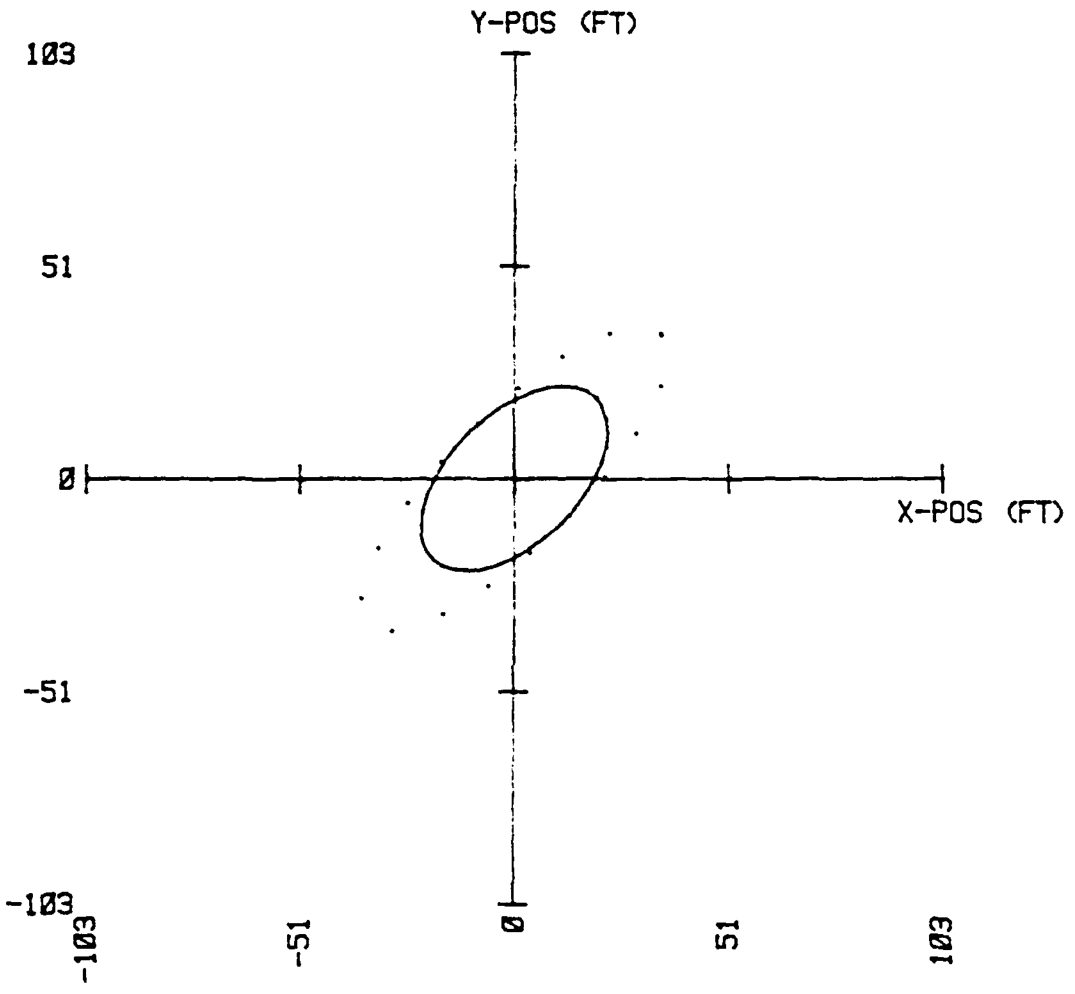
Figure 5.32 Filtered Track 4 Error Ellipses, σ_v=30, Reinitialize 5



SIGV=30 R=900 COVW=150

TIME	AREA(SQ FT)	LEG.	RESET
45	4383E+000	——	38
55	6852E+000	....	55
65	6495E+000	_ _ _	55

Figure 5.33 Filtered Track 4 Error Ellipses, $\sigma_v=30$, Reinitialize 5



```
SIGV=30  R=900  COVW=150       TIME  AREA(SQ FT)  LEG.  RESET
                                75   1348E+000   ——     69
                                85   2452E+000   ....   81
```

Figure 5.34 Filtered Track 4 Error Ellipses, σ_v=30, Reinitialize 5

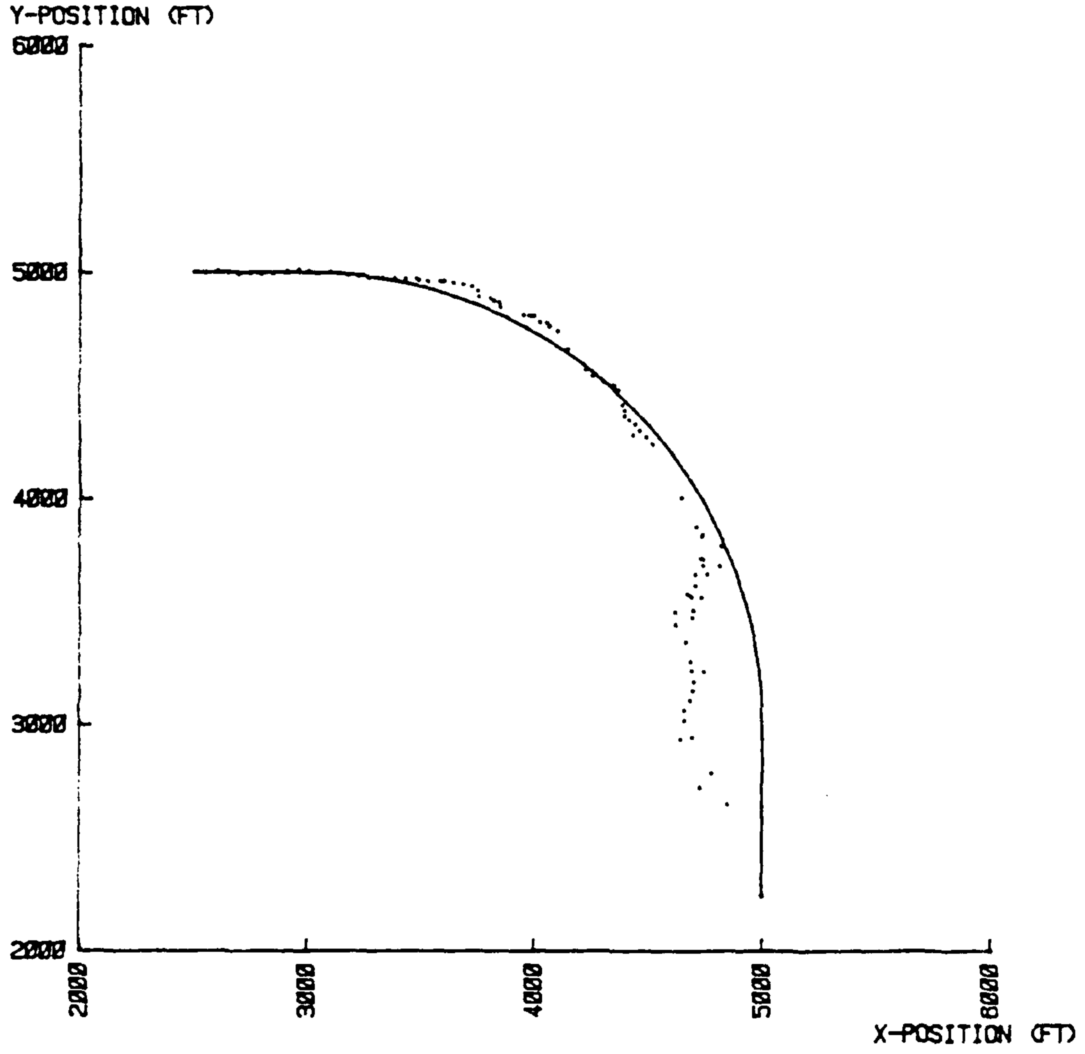


Figure 5.35 Filtered Track 4, σ_v=30, $\underline{R}$=900, σ_w^2=150, Reinitialize 7

VI. CONCLUSIONS AND RECOMMENDATIONS

A. ERROR ELLIPSE

The filter error ellipse proved useful as a tool for indicating filter performance. The information provided by the ellipse, particularly surface area changes, was used to make decisions concerning the alteration of the filter parameters. Several approaches for using the error ellipse were applied to both the linear and nonlinear tracking problem and the results are summarized below:

Procedure	Applicable Filter (Linear or Extended)	Comments
Statistics Window	Both	Useful in keeping the error ellipse current and responsive to present data. The ellipse reflects most recent data; old data is disregarded. Beneficial when making a decision concerning filter parameter modification. Normally a better indicator of filter convergence or divergence than without a statistics window.
Normalized Error Ellipse	Both	Aid in displaying error trends as target approaches coordinate axis or origin. Not practical in determining filter convergence/divergence due to rapid ellipse changes in vicinity of axes.

Reinitialize Filter- Increasing Ellipse Area	Linear- Set B(0) Extended-Set $\underline{P}(k+1/k) = 10^n \times \underline{P}_o$	Use increasing ellipse size as an indicator of divergence, and as a decision-making device to reinitialize filter. Particularly valuable later in track when gains and $\underline{P}(k+1/K)$ have settled out. More effective when used in conjunction with statistics window.
Error Ellipse Expansion/ Compression to vary COVW (Adaptive Q)	Extended	This technique of varying COVW based on error ellipse area increase or decrease is particularly useful in a tracking environment containing large variations in the random forcing input. The procedure is more effective when used in conjunction with a statistics window. Using this technique with filter reinitialization was unsuccessful.

B. COMPUTER PERFORMANCE

The HP-86 proved to be an extremely reliable computer with no downtime experienced during the 5 month period of operation. Full(Real) precision was used throughout the study providing 15 digit precision, which was more than adequate. The use of the Matrix ROM reduced program length by 90% and increased computing speed by a factor of about 6. Although not used in this study, a Statistics ROM would have undoubtedly further increased computing speed. For any further study using the HP-86, it is recommended that a Statistics ROM be procured.

It took approximately 2 seconds for each incremental time measurement data to be sequenced through the filter equations, both for the linear and nonlinear case. This computing time also included all statistics computations. With an incoming data rate of 1 set of measurement data per second, this computing speed is not sufficient for on-line processing. As previously mentioned, the Naval Underwater Tracking Range receives a series of 4 measurement times sequentially every 1.31 seconds. The range's three-dimensional tracking problem will necessarily involve more than the 4 state variables used in this study. Hence greater matrix dimensions resulting in longer computing times can be expected.

The HP-86 CRT graphics were used extensively to provide error ellipse plots during the tracking runs. Using a "no frills" approach to plotting, i.e. plotting without x-y axis or labelling, it took approximately 2.5 seconds per ellipse plot. The ellipse plotting routine used involved sines and cosines, plotted point by point in 30 degree increments for a total of 360 degrees. This method was somewhat slow. Had there been available a graphics program that would sketch in the ellipse around the intersected major and minor axis, the graphics presentation could have been speeded up. But since ellipse plotting for every 3 to 5 increments of time provided sufficient "real-time" information, the time of 2.5 seconds per plot was tolerable.

Summarizing, the HP-86 could be used to compute statistics and provide graphics in the real-time underwater tracking environment, if the graphics were required not more often than 3 to 5 seconds. However, before the HP-86 can be considered feasible for real-time Kalman filter processing, more investigation is needed in finding ways to speed up computer processing time such as parallel processing, additional use of manufacturer-provided ROMs and machine language programming.

APPENDIX A
TRACK GENERATION

1. TRACK THREE

Target movement is in the x-y plane with the tracking sensor located at the origin of the cartesian axes. The target follows a parabolic track (see Figure A-1) at a constant speed of 200 feet per second.

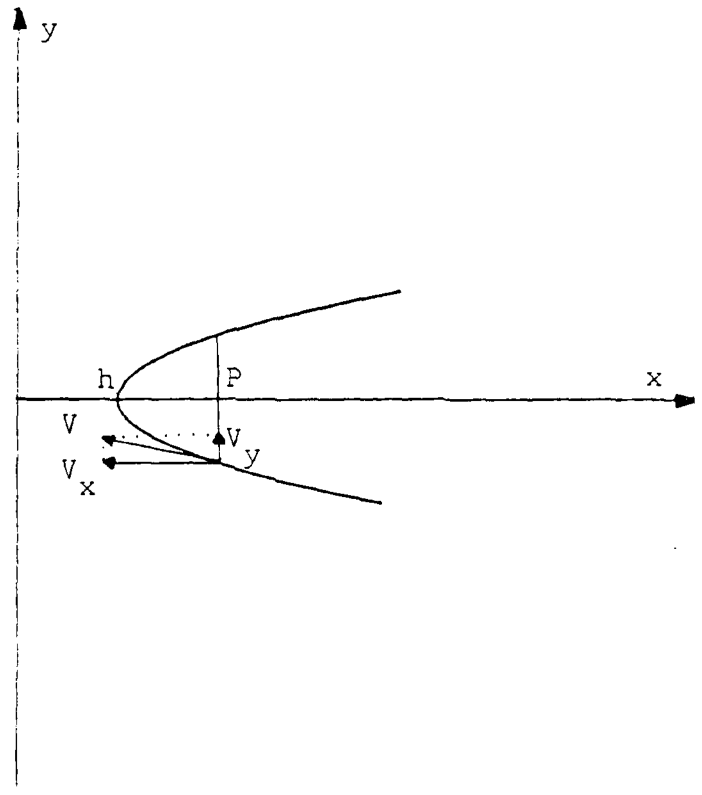


Figure A-1 Track 3

The parabolic equation is:

$$y^2 = 4p(x-h)$$

where p = 1000 and h = 1000, resulting in:

$$y^2 = 4000(x-1000)$$

Initial target location is (x,y) = (8000, 5291.5). Target x-direction velocity is given by:

$$v_x = v \cos(\text{Angle}) \qquad \text{(A-1)}$$

and target y-direction velocity is:

$$v_y = v \sin(\text{Angle}) \qquad \text{(A-2)}$$

where the Angle is obtained from:

$$\text{Angle} = \tan^{-1}(\frac{\Delta y}{\Delta x})$$

and v is obtained from:

$$v = (v_x^2 + v_y^2)^{1/2}$$

where the argument of the inverse tangent is the slope of a small increment (less than 1 second) at each successive data point. Using a sampling period of one second, the data points (x,y) can be obtained from:

$$x(k+1) = x(k) + v_x(k)$$

$$y(k+1) = y(k) + v_y(k)$$

Table A-1 gives the numerical values for the four states.

2. TRACK FOUR

Target movement simulates a 30-knot torpedo (velocity 50 feet per second) at a constant depth. The target's initial position is (x,y) = (3250, 5000), and it is moving in the v_x direction with $v_y=0$. (See Figure A-2.) The target remains on a straight course for 15 seconds, and then executes a 90 degree turn and travels in the -y direction at $v_y = -50$ ft/sec. The trajectory of the target turn is described by a 90 degree arc of a circle of radius, R = 1000, with the circle centered at (x,y) = (4000, 4000). The 90 deg arc will be traversed in:

$$2\pi R/4v \text{ sec.} = 10\pi \text{ sec.} (\text{where } v = 50 \text{ ft/sec})$$

TABLE A-1

NUMERICAL VALUES OF STATES FOR TRACK THREE

K	X	X-VEL	Y	Y-VEL
1	8000.00	-186.90	5291.50	-71.15
2	7813.10	-186.93	5220.38	-71.13
3	7626.17	-186.60	5148.27	-71.98
4	7439.58	-186.25	5075.26	-72.87
5	7253.33	-185.89	5001.33	-73.79
6	7067.44	-185.51	4926.43	-74.74
7	6881.93	-185.11	4850.54	-75.73
8	6696.82	-184.68	4773.60	-76.76
9	6512.14	-184.24	4695.59	-77.83
10	6327.90	-183.76	4616.45	-78.94
11	6144.14	-183.26	4536.14	-80.09
12	5960.87	-182.73	4454.60	-81.30
13	5778.14	-182.17	4371.79	-82.56
14	5595.98	-181.57	4287.65	-83.87
15	5414.41	-180.92	4202.10	-85.24
16	5233.49	-180.24	4115.09	-86.68
17	5053.25	-179.51	4026.54	-88.19
18	4873.74	-178.72	3936.37	-89.78
19	4695.02	-177.87	3844.49	-91.44
20	4517.15	-176.96	3750.81	-93.19
21	4340.19	-175.97	3655.24	-95.04
22	4164.22	-174.91	3557.65	-97.00
23	3989.31	-173.74	3457.93	-99.06
24	3815.57	-172.48	3355.93	-101.25
25	3643.09	-171.09	3251.52	-103.57
26	3472.00	-169.57	3144.52	-106.05
27	3302.43	-167.89	3034.75	-108.68
28	3134.54	-166.04	2922.01	-111.50
29	2968.50	-163.98	2806.06	-114.51
30	2804.52	-161.67	2686.65	-117.74
31	2642.85	-159.09	2563.47	-121.21
32	2483.76	-156.17	2436.20	-124.94
33	2327.59	-152.86	2304.42	-128.98
34	2174.74	-149.07	2167.70	-133.33
35	2025.66	-144.72	2025.50	-138.05
36	1880.95	-139.67	1877.18	-143.15
37	1741.28	-133.78	1721.95	-148.67
38	1607.50	-126.83	1558.85	-154.64
39	1480.67	-118.59	1386.61	-161.05
40	1362.08	-108.70	1203.46	-167.88
41	1253.38	-96.73	1006.73	-175.05
42	1156.65	-82.01	791.58	-182.41
43	1074.64	-63.44	546.41	-189.67
44	1011.20	-37.24	211.63	-196.50
45	1000.00	0.00	0.00	-200.00

TABLE A-1 (CONT.)

K	X	X-VEL	Y	Y-VEL
46	1011.20	10.57	-211.63	-199.72
47	1021.77	25.14	-295.07	-198.41
48	1046.90	35.82	-433.13	-196.77
49	1082.72	48.89	-575.23	-193.93
50	1131.61	61.85	-725.57	-190.20
51	1193.46	74.49	-879.69	-185.61
52	1267.95	86.36	-1035.28	-180.39
53	1354.31	97.25	-1190.48	-174.77
54	1451.56	107.04	-1343.96	-168.94
55	1558.60	115.75	-1494.80	-163.10
56	1674.35	123.43	-1642.38	-157.37
57	1797.78	130.16	-1786.37	-151.35
58	1927.94	136.06	-1926.60	-146.58
59	2064.01	141.24	-2063.01	-141.61
60	2205.24	145.78	-2195.67	-136.92
61	2351.02	149.78	-2324.67	-132.54
62	2500.80	153.31	-2450.14	-128.43
63	2654.11	156.45	-2572.25	-124.60
64	2810.56	159.24	-2691.14	-121.01
65	2969.80	161.73	-2806.99	-117.66
66	3131.52	163.97	-2919.95	-114.52
67	3295.49	165.98	-3030.17	-111.58
68	3461.47	167.80	-3137.81	-108.82
69	3629.27	169.46	-3243.01	-106.23
70	3798.73	170.96	-3345.88	-103.79
71	3969.69	172.34	-3446.56	-101.49
72	4142.03	173.60	-3545.15	-99.32
73	4315.63	174.76	-3641.77	-97.26
74	4490.38	175.82	-3736.51	-95.32
75	4666.21	176.81	-3829.47	-93.48
76	4843.02	177.73	-3920.72	-91.73
77	5020.74	178.57	-4010.36	-90.06
78	5199.32	179.36	-4098.45	-88.48
79	5378.68	180.10	-4185.06	-86.97
80	5558.78	180.79	-4270.26	-85.53
81	5739.57	181.44	-4354.11	-84.15
82	5921.01	182.04	-4436.67	-82.83
83	6103.05	182.61	-4517.99	-81.57
84	6285.66	183.14	-4598.11	-80.36
85	6468.80	183.65	-4677.09	-79.20
86	6652.45	184.13	-4754.98	-78.09
87	6836.58	184.58	-4831.80	-77.01
88	7021.16	185.00	-4907.61	-75.98
89	7206.16	185.41	-4982.43	-74.99
90	7391.57	185.79	-5056.31	-74.03

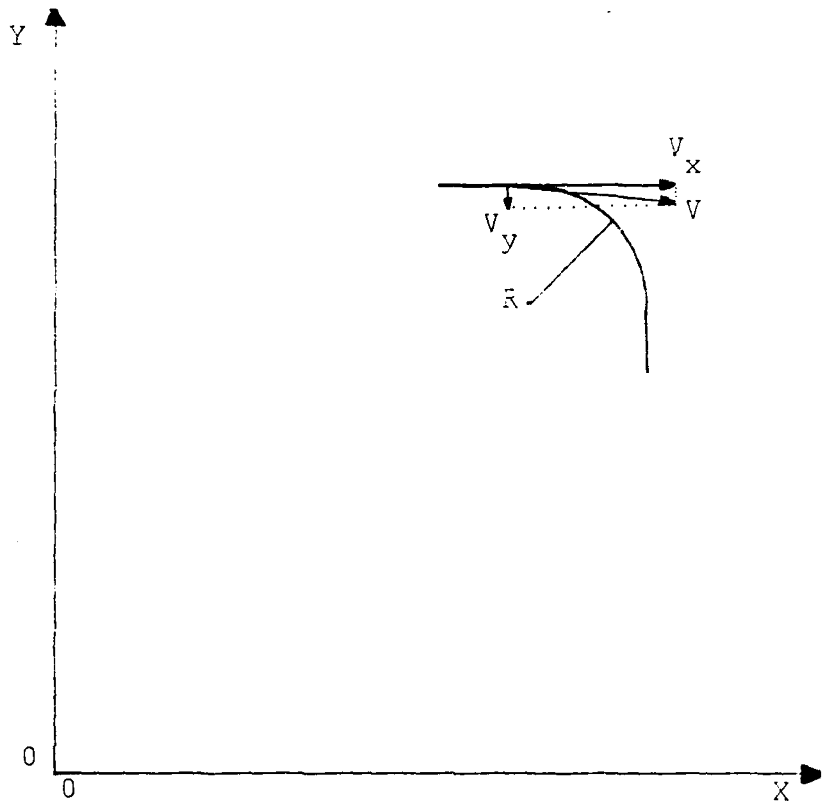


Figure A-2 Track 4

So each second:

$$\frac{2\pi/4}{10\pi} = .05 \text{ radians will be traversed}$$

Using the trigonometric identity:

$$\sin^2(A) + \cos^2(A) = 1$$

And the equation for a circle:

$$x^2 + y^2 = c^2$$

It follows that the arc of Figure A-2 is described by:

$$x(k) = 4000 + 1000\sin(.05k)$$

$$y(k) = 4000 + 1000\cos(.05k)$$

where the angle argument is in radians and k=0,1,2,...31 seconds. The velocities v_x and v_y can be obtained from:

$$v_x(k) = 50\cos(.05k)$$

$$v_y(k) = -50\sin(.05k)$$

using the same angle argument.

The track values for the track arc are contained in Table A-2.

TABLE A-2

NUMERICAL VALUES OF STATES FOR TRACK FOUR

K	X	X-VEL	Y	Y-VEL
12	3049.99	49.98	4999.38	-1.25
13	3099.96	49.94	4997.50	-2.50
14	3149.86	49.86	4994.38	-3.75
15	3199.67	49.75	4990.01	-4.99
16	3249.35	49.61	4984.40	-6.23
17	3298.88	49.44	4977.54	-7.47
18	3348.22	49.24	4969.45	-8.71
19	3397.34	49.00	4960.13	-9.93
20	3446.21	48.74	4949.59	-11.16
21	3494.81	48.45	4937.82	-12.37
22	3543.09	48.12	4924.85	-13.58
23	3591.04	47.77	4910.67	-14.78
24	3638.62	47.38	4895.30	-15.97
25	3685.80	46.97	4878.75	-17.14
26	3732.55	46.53	4861.02	-18.31
27	3778.84	46.05	4842.12	-19.47
28	3824.64	45.55	4822.08	-20.62
29	3869.93	45.02	4800.89	-21.75
30	3914.68	44.46	4778.59	-22.87
31	3958.85	43.88	4755.17	-23.97
32	4002.43	43.27	4730.65	-25.06
33	4045.37	42.63	4705.05	-26.13
34	4087.67	41.96	4678.38	-27.19
35	4129.28	41.27	4650.67	-28.23
36	4170.19	40.55	4621.93	-29.25
37	4210.37	39.80	4592.17	-30.26
38	4249.79	39.04	4561.41	-31.24
39	4288.44	38.24	4529.68	-32.21
40	4326.27	37.42	4497.00	-33.16
41	4363.28	36.58	4463.38	-34.08
42	4399.43	35.72	4428.84	-34.99
43	4434.71	34.84	4393.41	-35.87
44	4469.10	33.93	4357.11	-36.73
45	4502.56	33.00	4319.97	-37.56
46	4535.09	32.05	4281.99	-38.38
47	4566.65	31.08	4243.22	-39.17

TABLE A-2 (CONT.)

K	X	X-VEL	Y	Y-VEL
48	4597.24	30.09	4203.67	-39.93
49	4626.83	29.08	4163.37	-40.67
50	4655.40	28.06	4122.34	-41.39
51	4682.94	27.02	4080.60	-42.07
52	4709.43	25.95	4038.20	-42.74
53	4734.85	24.88	3995.14	-43.37
54	4759.18	23.79	3951.46	-43.98
55	4782.41	22.68	3907.19	-44.56
56	4804.54	21.56	3862.35	-45.11
57	4825.53	20.42	3816.97	-45.64
58	4845.38	19.28	3771.09	-46.13
59	4864.08	18.12	3724.72	-46.60
60	4881.61	16.95	3677.89	-47.04
61	4897.97	15.77	3630.64	-47.45
62	4913.14	14.58	3583.00	-47.83
63	4927.12	13.37	3535.00	-48.18
64	4939.89	12.17	3486.66	-48.50
65	4951.45	10.95	3438.01	-48.79
66	4961.79	9.73	3389.10	-49.04
67	4970.90	8.50	3339.93	-49.27
68	4978.78	7.26	3290.56	-49.47
69	4985.43	6.03	3241.01	-49.64
70	4990.83	4.78	3191.30	-49.77
71	4994.99	3.54	3141.47	-49.87
72	4997.90	2.29	3091.56	-49.95
73	4999.57	1.04	3041.59	-49.99
74	4999.98	-.21	2991.59	-50.00

APPENDIX B

COMPUTER PROGRAM EXPLANATION

1. LINKAL

The LINKAL program computes the filter gains, GAIN(4,2), for the 4-state system and stores the gains in "LINGAIN .STORAG". The theoretical covariance of error matrix, PKK(4,4), is also computed and stored in "LINCOV.STORAG". Several different sets of gain and covariance values were computed and stored for various values of measurement noise, RMAT(2,2), and random forcing noise, COVW(2,2).

2. LINEST

The LINEST program retrieves the appropriate gain schedule from storage and computes the optimal estimate, XHAT(4,1), and the optimal one-step prediction, XHK1K(4,1). The following capabilities are contained in the LINEST program:

a. Gating Scheme

If the absolute difference (DIFF) between the one-step prediction, XHK1K and the noisy track value, ZMAT, is greater than the three-sigma gate, then the GAIN matrix is disregarded and XHAT is set equal to XHK1K.

b. Track Noise

By setting NOITRAK = 1, the random number generator, RND, and the resulting simulated noise produced, V1 and V2, are bypassed and the track values corrupted with noise, ZMAT, are retrieved from data file "NOITRAK.STORAG". If NOITRAK=1, then the random number generator is "reseeded" for each program run, producing a different set of noise values resulting in a unique noise corrupted track for each run.

c. <u>Statistics Window</u>

When WINDOW is set to 0, the filter state statistics are computed after each Monte Carlo run. The error mean, variance and position covariance are computed. If WINDOW is set to an integer, I, such that max $k > I > 0$, the statistics will be computed based on the data compiled during the last I iterations of the simulation. If, for example, WINDOW=10, the computation of statistics will be based on the data obtained during the last ten iterations of k, and all previous data is disregarded.

d. <u>Error Normalization</u>

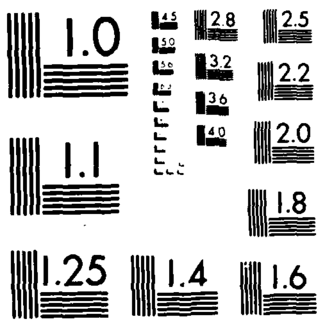
By setting NORM = 1, the error (ERR), which is defined as the difference between the true track value (TRAK) and the estimate (XHAT), is normalized. For all other values of NORM the relative error is used in computing the statistics.

e. <u>Reinitializing Gains</u>

The program has the option of reinitializing the gains to G(0). This can be done by setting HIGH to an integer I, such that max $k > I > 0$. If the surface area of the ellipse

AD-A143 630   DEVELOPMENT OF REAL-TIME ERROR ELLIPSES AS AN INDICATOR   2/2
OF KALMAN FILTER PERFORMANCE(U) NAVAL POSTGRADUATE
SCHOOL MONTEREY CA J JAROS MAR 84

UNCLASSIFIED   F/G 9/2   NL


1.0
4.5
5.0
5.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

increases I consecutive times, indicating filter divergence, the gains are reinitialized to G(0). If reinitializing gains is not desired as an option, then HIGH is set to some arbitrary large number greater than max k.

3. EXTKF

The EXTKF program computes the optimal estimate, XHAT, and the optimal one-step prediction, XHK1K, for the 4-state nonlinear tracking problem. The EXTKF program has the options: gating scheme, track noise, statistics window, and error normalization as described for the LINEST program. EXTKF has the following additional options:

a. Reinitializing the Filter

The program has the option of reinitializing the covariance of one-step prediction error matrix, PK1K, by setting it to $10^n x \underline{P}_o$, where n is zero or some small integer. If HIGH is set to an integer I, such that max k>I>0, and the surface area of the ellipse increases I consecutive times, then the PK1K matrix will be reset to $10^n x \underline{P}_o$.

b. Adaptive Q

EXTKF has the option of automatically increasing or decreasing the state excitation matrix, $\underline{Q}$, under certain conditions by changing the value of the covariance of excitation noise, COVW. INCREASE and DECREASE are set to integers I and J respectively, such that max k>I,J>0. If the surface area of the error ellipse increases I consecutive times, COVW is doubled or increased by some factor. Conversely, if the surface area of the error ellipse decreases J consecutive times, COVW is halved or decreased by some factor. If COVW is changed, increased for example, the value of COVW can be changed again later in the run, if the criteria for either increasing or decreasing is met. If the adaptive $\underline{Q}$ is not desired then INCREASE and DECREASE are set to some large number greater than k.

APPENDIX C

PROGRAM LISTING

```
10 !linkal - this program computes the gain schedule and the
    error covariance matrix schedule for the linear kal. filter.
20 option base 1
30 real phi(4,4),cmat(2,4),gamma(4,2),covw(2,2),rmat(2,2)
40 real pk1k(4,4),imat(4,4),phit(4,4),gammat(2,4),qmat(4,4)
50 real temp1(2,4),temp2(4,2),temp3(2,2),gtemp(2,2)
60 real gtempin(2,2),temp4(4,2),gain(4,2),temp5(4,4)
70 real temp6(4,4),pkk(4,4),temp7(4,4),temp8(4,4),cmatt(4,2)
80 assign# 1 tc "lingain5 .storag"
90 assign# 2 tc "linccv5 .storag"
100 mat read phi
110 data 1,1,0,C,0,1,0,C,0,0,1,1,0,0,0,1
120 mat read cmat
130 data 1,0,C,C,0,C,1,C
140 mat read gamma
150 data .5,0,1,0,0,.5,C,1
160 mat read covw
170 10C,C,0,10C
180 mat read rmat
190 data 2500,C,0,25C0
200 mat read pk1k
210 data 10e9,C,0,0,C,1Ce9,0,0,0,0,10e9,0,0,0,0,10e9
220 !************* matrix transpose **********************
230 mat cmatt = trn(cmat)
240 mat phit = trn (phi)
250 mat gammat = trn (gamma)
260 mat imat = idn
270 mat temp1 = covw*gammat
280 mat qmat = gamma*temp1
290 !************************************************************
300 ! compute gain schedule, gain() and covariance matrix, p(k/k)
    using equations (2-10),(2-11), and (2-12).
310 for k = 0 tc 60
320 disp "k= ";k
330 mat temp2 = pk1k*cmatt
340 mat temp3 = cmat*temp2
350 mat gtemp = temp3 + rmat
360 mat gtempin = inv (gtemp)
370 mat temp4 = cmatt*gtempin
380 mat gain = pk1k*temp4
390 !****************** print gain matrix **********************
400 disp "gain"
```

```
10 !lipest
20 !this program computes the estimated track for the linear kal-
    man filter using gain schedule and covariance matrix computed
30 ! off-line.  it also computes statistics and covariance of
40 ! error matrix.
50 cption base 1
60 real gain(4,2),cmat(2,4),phi(4,4),xhk1k(4,1),temp1(2,1),errsq(4)
70 real pkk(4,4),trak(4,1),zmat(2,1),temp2(2,1),temp3(4,1),ekk(3)
80 real xhat(4,1),diff(2),err(4),ssum(4),mean(4),sumsq(4),var(4)
90 real covmat(4,4),meansq(4),meanksq(4),mnprod(4,4),tprod(4,4)
100 real prod(4,4),ccvmax(90),wssum(4,90),wsumsq(4,90),wtprod(90)
110 real wprcd(90),werr(4,90),werrsq(4,90),wmean(4,90),wmeansq(4,90)
120 real wmeanksq(4,90),wvar(4,90),wcovmat(3,90),wmnprod(90)
130 n = 4              ! number of states
140 s = 1              ! x-state of error ellipse
150 t = 3              ! y-state of error ellipse
160 noise = 30C
170 r = 20000
180 sigw = 10
190 norm = 0           !"1" if error is tc normalized.
200 pltarea = C        !"1" if only area is printed out, i.e. no ellipse.
210 pl = 1             !"1" if plotting cn screen, otherwise on plotter.
220 statplt = 0        !"1" if ellipse plotting from previously computed
                        statistics.
230 noitrak = 1        !"1" if using previous noisy  track from "noitrak".
240 high = 10          ! indicate # of times ellipse increases consecu-
                         tively before resetting gains.
250 windcw = 1C        !"0" if no statistics window, otherwise indicate
                         the length of the windcw.
260 first = 5          ! indicate the first iteratin of k to be plotted.
270 nr = 10            ! plct the error ellipse every "nr" iterations of k
280 chg = 3            ! clear the screen every "chg" plots.
290 pt = chg + .5
300 sscale = 0         ! "1" if scale is to be changed after clearing.
310 e = 4              ! 1/e is the scale size of the ellipse.
320 stp = 10           ! iterate the plct every "stp" degrees
330 cl = 0
340 count = 0
350 larger = 0
360 reset = 0
370 flag = 0
380 zz = 0
```

```
390 llarge = 0
400 assign# 1 tc "estdata .storag"
410 assign# 2 to "datatrak7, drive0"
420 assign# 3 tc "lingain3 .storag"
430 assign# 4 to "trkest3 .storag"
440 assign# 5 tc "linerr3 .storag"
450 assign# 6 to "statlin3 .storag"
460 assign# 7 tc "lincov3 .storag"
470 assign# 8 tc "noitrak7c .storag"
480 read# 1 ; phi(),cmat()
490 read# 2 ; xhk1k()
500 randcmize
510 deg
520 mat ssum = zer
530 mat sumsq = zer
540 mat prod = zer
550 for k = 1 to 70
560   if statplt=1 then 2080
570   read# 7 ; pkk()
580 !****************** compute 3-sigma gate *************************
590   maxpk = pkk(1,1)
600   for i = 2 to 4
610      if pkk(i,i)>maxpk then maxpk = pkk(i,i)
620   next i
630   gate = 3*sqr (maxpk + r)
640   if reset = 0 then 700
650      assign# 3 tc *
660      assign# 3 tc "lingain3 .stcrag"
670      assign# 7 tc *
680      assign# 7 tc "lincov3 .storag"
690      reset = 0
700   1 = k - 1
710   mat temp1 = cmat*xhk1k
720   read# 2 ; trak()
730   if noitrak = 1 then 830
740 !*****************************************************************
750 !              ccmpute track corrupted with noise
760 !*****************************************************************
770   v1 = ncise*2*(rnd - .5)
780   v2 = ncise*2*(rnd - .5)
790   zmat(1,1) = trak(1,1) + v1
800   zmat(2,1) = trak(2,1) + v2
```

```
810   print# 8 ; zmat()
820   goto 840
830   read# 8 ; zmat()
840 !*********** compute the difference for the gate test ***********
850   diff(1) = abs(xhk1k(1,1) - zmat(1,1))
860   diff(2) = abs(xhk1k(3,1) - zmat(2,1))
870 !***************************************************************
880 !                compute xhat(k/k)
881 !***************************************************************
890   read# 3 ; gain()
900   if diff(1)>gate or diff(2)>gate then mat xhat=xhk1k @ goto 980
910   mat temp2 = zmat - temp1
920   mat temp3 = gain*temp2
930   mat xhat = xhk1k + temp3
940   goto 980
950 !**************** print estimated track ***************************
960   disp "xhat(";l;")"
970   mat disp xhat,
980   print# 4 ; xhat()
990 !***************************************************************
1000 !                compute xhat(k+1/k)
1001 !**************************************************************
1010  mat xhk1k = phi*xhat
1020 !************** compute statistics *******************************
1030 !          compute running sum of error
1031 !**************************************************************
1040  mat err = trak - xhat
1050  if norm<> 1 then 1100
1060  for i = 1 to 4
1070      if trak(i,1)=0 then trak(i,1)=1
1080  next i
1090  mat err = err/trak
1100  print# 5 ; err()
1110  if window>0 then 1420
1120  mat ssum = err + ssum
1130 !********** compute the mean of the error **********************
1140  zz = zz + 1
1150  mat mean = (1/zz)*ssum
1160 !**********  compute the variance of the error *****************
1170  mat errsq = err.err
1180  mat sumsq = errsq + sumsq
1190  mat meansq =  (1/zz)*sumsq
```

```
1200  mat meanksq = mean.mean
1210  mat var = meansq - meanksq
1220 ! ************ compute the running product mean ****************
1230  for i = 2 to n
1240      for j = 1 to i-1
1250          tprod(i,j) = err(i)*err(J)
1260          prod(i,j) = tprod(i,j) + prod(i,j)
1270          mnprod(i,j) = prod(i,j)/k
1280      next j
1290  next i
1300 !*****  compute the diagonal terms of the cov of error matrix ***
1310  for i = 1 to n
1320      covmat(i,i) = var(i)
1330  next i
1340 ! *** compute the off-diagonal terms of the cov of error matrix**
1350  for i = 2 to n
1360      for j = 1 to i-1
1370          covmat(i,j) = mnprod(i,j) - mean(i)*mean(j)
1380          covmat(j,i) = covmat(i,j)
1390      next j
1400  next i
1410  goto 2090
1420  for i = 1 to 4
1430      werr(i,k) = err(i)
1440  next i
1450  if k = 1 then 1510
1460  if k>window then 1830
1470  for i = 1 to 4
1480      wssum(i,k) = werr(i,k) + wssum(i,k-1)
1490  next i
1500  goto 1540
1510  for i = 1 to 4
1520      wssum(i,k) = werr(i,k)
1530  next i
1540 !***************** compute the mean of error ******************
1550  for i = 1 to 4
1560      wmean(i,k) = 1/k*wssum(i,k)
1570  next i
1580 ! *************** compute the variance of error ****************
1590  for i = 1 to 4
1600      werrsq(i,k) = werr(i,k)*werr(i,k)
1610  next i
```

```
1620  if k = 1 then 1670
1630  for i = 1 to 4
1640      wsumsq(i,k) = werrsq(i,k) + wsumsq(i,k-1)
1650  next i
1660  goto 1700
1670  for i = 1 to 4
1680      wsumsq(i,k) = werrsq(i,k)
1690  next i
1700  for i = 1 to 4
1710      wmeansq(i,k) = 1/k*wsumsq(i,k)
1720      wmeanksq(i,k) = wmean(i,k)*wmean(i,k)
1730      wvar(i,k) = wmeansq(i,k) - wmeanksq(i,k)
1740  next i
1750 !************  compute the running mean product *****************
1760  wtprod(k) = werr(s,k)*werr(t,k)
1770  if k = 1 then 1800
1780     wprod(k) = wtprod(k) + wprod(k-1)
1790     goto 1810
1800  wprod(k) = wtprod(k)
1810  wmnprod(k) = 1/k*wprod(k)
1820  goto 2050
1830 ! **************** computing the window statistics **************
1840  for i = 1 to 4
1850      wssum(i,k) = werr(i,k) + wssum(i,k-1) - werr(i,k-window)
1860  next i
1870 !************** compute the mean of error **********************
1880  for i = 1 to 4
1890      wmean(i,k) = 1/window*wssum(i,k)
1900  next i
1910 ! ************ compute the variance of error *******************
1920  for i = 1 to 4
1930      werrsq(i,k) = werr(i,k)*werr(i,k)
1940      wsumsq(i,k) = werrsq(i,k)+wsumsq(i,k-1) - werrsq(i,k-window)
1950  next i
1960  for i = 1 to 4
1970      wmeansq(i,k) = 1/window*wsumsq(i,k)
1980      wmeanksq(i,k) = wmean(i,k)*wmean(i,k)
1990      wvar(i,k) = wmeansq(i,k) - wmeanksq(i,k)
2000  next i
2010 !*************** compute the running product mean ***************
2020  wtprod(k) = werr(s,k)*werr(t,k)
2030  wprod(k) = wtprod(k) + wprod(k-1) - wtprod(k-window)
```

```
2040  wmnprod(k) = 1/windcw*wprod(k)
2050  covmat(s,s) = wvar(s,k)
2060  covmat(t,t) = wvar(t,k)
2070  covmat(s,t) = wmnprod(k) - wmean(s,k)*wmean(t,k)
2080  if statplt=1 then read# 6 ; covmat(s,s),covmat(t,t),covmat(s,t) &
      goto 2100
2090  print# 6 ; covmat(s,s),covmat(t,t),covmat(s,t)
2100  covmax(k) = covmat(s,s)*covmat(t,t)
2110  if k<2 then 2210
2120  if covmax(k)>covmax(k-1) then larger = larger+1 else larger=0
2130  if larger<high then 2210
2140    reset = 1
2150    flag = 1
2160    mat ssum = zer
2170    mat sumsq = zer
2180    mat prod = zer
2190    zz = 0
2200    kreset = k
2210 !*************** plot the error ellipse ***************************
2220  if k<first then 3110
2230  if k=first or (k-first) mod nr = 0 then 2240 else 3110
2240  count = count + 1
2250  ekk(1) = covmat(s,s)
2260  ekk(2) = covmat(t,t)
2270  ekk(3) = covmat(s,t)
2280  lx = ekk(1)
2290  ly = ekk(2)
2300  mmax = max (lx,ly)
2310  mmin = min (lx,ly)
2320  if mmin <= 0 then 3040
2330  if mmax/mmin<10000 then 2350
2340  if mmax = lx then llarge = 1 else llarge = 2
2350  if ekk(3)<> 0 then 2380
2360   theta = 0
2370  goto 2450
2380  if ekk(1)<> ekk(2) then 2410
2390   theta = 0
2400  goto 2450
2410  theta = .5*atn (2*ekk(3)/(ekk(1)-ekk(2)))
2420  lx = (ekk(1) + ekk(2))/2 + ekk(3)/sin(2*theta)
2430  ly = (ekk(1) + ekk(2))/2 - ekk(3)/sin(2*theta)
2440  if lx<0 or ly<0 then 3110
```

```
2450  a = sqr (lx)
2460  b = sqr (ly)
2470  area = 3.1416*a*b
2480  if pltarea = 1 then 3100
2490  if k = first then 2520
2500  if count = 1 and sscale = 1 then 2520
2510  goto 2550
2520  smax = max (a,b)
2530  pscale = e*smax
2540  nscale = -(e*smax)
2550  if pl=1 then plotter is 1 else plotter is 805
2560  if cl<> 0 then 2600
2570  pt = chg + .5
2580  gclear
2590  if pl=0 then limit 28,241,35,184
2600  if pl=1 then locate 5,145,16,90 else locate 30,108,18,96
2610  scale nscale,pscale,nscale,pscale
2620  if cl<>0 then 2720
2630  move pscale,.1*nscale
2640  label "x-position(ft)"
2650  move .2*nscale,1.05*pscale
2660  label "y-position(ft)"
2670  move .9*pscale,1.05*pscale
2680  label "time";" area(sq ft)";" legend"
2690  move .4*nscale,1.3*nscale
2700  label using 2710 ; "sigw=";sigw;"  sigv=";noise;"  r=";r
2710  image k,k,k,k,k,k
2720  if sscale<>1 and k<>first then 2750
2730  if cl = 0 then laxes pscale/2,pscale/2,0,0
2740  goto 2760
2750  if pl<> 1 and cl = 0 then laxes pscale/2,pscale/2,0,0
2760  if pl=1 then move .8*pscale,pt/(chg + 1)*pscale @ goto 2780
2770  move .9*pscale,pt/(chg +1)*pscale
2780  if llarge = 0 then 2830
2790  if llarge = 1 then label using 2800; "  ";k;"  ";area;"  x-axis"
      else label using 2800; "  ";k;"  ";area;"  y-axis"
2800  image 2a,dd,2a,4de,8a
2810  llarge = 0
2820  goto 2880
2830  if count = 1 then label using 2870; "  ";k;"  ";area;"  ----"
2840  if count = 2 then label using 2870; "  ";k;"  ";area;"  ...."
2850  if count = 3 then label using 2870; "  ";k;"  ";area;" - - -"
```

```
2860  if count = 4 then label using 2870: "  ";k;"  ";area;"  ____"
2870  image 2a,dd,2a,4de,6a
2880  if flag = 0 then 2920
2890  move 1.7*nscale,pt/(chg+1)*pscale
2900  label "reset at k=";kreset
2910  flag = 0
2920  move 0,0
2930  if count = 2 then line type 3
2940  if count = 3 then line type 5
2950  if count = 4 then line type 1
2960  for ang = 0 to 360 step stp
2970      xx = cos(ang)*a
2980      yy = sin(ang)*b
2990      x1 = xx*cos(theta) - yy*sin(theta)
3000      y1 = yy*cos(theta) + xx*sin(theta)
3010      rplot x1,y1
3020  next ang
3030  goto 3050
3040  disp "k=";k;"  no ellipse plotted.  out of scale."
3050  cl = cl + 1
3060  pt = pt - .4
3070  if cl = chg then cl = 0
3080  if count = chg then count = 0
3090  goto 3110
3100  disp "k=";k;area
3110 next k
3120 assign# 1 to *
3130 assign# 2 to *
3140 assign# 3 to *
3150 assign# 4 to *
3160 assign# 5 to *
3170 assign# 6 to *
3180 assign# 7 to *
3190 assign# 8 to *
3200 end
```

```
10   ! extkf - this program computes gain schedule, covariance matrix
20   ! and track estimates for 4 state tracking problem.
30   option base 1
40   deg
50   real hmat(1,4),hmatt(4,1),pk1k(4,4),pkk(4,4),rmat(1,1),trak(4,1)
60   real imat(4,4),xhk1k(4,1),phi(4,4),phit(4,4),temp1(4,1),temp2(1,1)
70   reAl TEMP3(1,1),temp3in(1,1),temp4(4,1),gain(4,1),temp5(4,4)
80   real temp6(4,4),zmat(1,1),chk1k(1,1),temp7(1,1),temp8(4,1)
90   real xhat(4,1),temp9(4,4),temp9a(4,4),gama(4,2),gamat(2,4)
100  real covw(2,2),tempq(2,4),qmat(4,4),err(4),ssum(4),mean(4)
110  real sumsq(4),errsq(4),covmat(4,4),ekk(3),meansq(4),var(4)
120  real meanksq(4),mnprod(4,4),tprod(4,4),prod(4,4),perr(3)
130  real wssum(4,70),wsumsq(4,70),wtprod(70),wprod(70),werr(4,70)
140  real werrsq(4,70),wmean(4,70),wmeansq(4,70),wmeanksq(4,70)
150  real wvar(4,70),wcovmat(3,70),wmnprod(70),ccov(70)
160  real reset(4,4)
170  randomize
180  assign# 1 tc "datatrak8 .storag"
190  pltarea = 1     !"1" if only area is being computed, i.e. no
                       ellipse plotting.
200  pl = 1          ! "1" if plot on crt screen, otherwise on plotter.
210  statplt = 0     !"1" if plotting previously computed statistics.
220  n = 4           ! number of states.
230  s = 1           ! x-state of error ellipse.
240  t = 3           ! y-state of error ellipse.
250  pplot = 0       ! "1" if pkk error ellipse; otherwise statistics
                       ellipse.
260  start = 1       ! "start" is the iteration of k to begin keeping
                       statistics.
270  high = 70       ! last k to be computed (>start and <91).
280  window = 10     ! "0" if no statistics window, otherwise indicate
                       the length of the window. (pplot must equal 0).
300  first = 10      ! first iteration of k to be plotted (greater than
                       or equal to start).
320  chg = 3         ! clear the screen every "chg" plots.
330  sscale = 0      ! "1" if scale is to be changed after clearing.
340  nr = 10         ! plot error ellipse every "nr" iterations of k.
350  norm = 0        ! "1" if error is to be normalized.
360  noitrak = 1     ! "1" if using previously determined noisy track.
370  stp = 10        ! iterate plot every "stp" degrees.
380  noise = 30      ! standard deviation of noise.
390  covw(1,1) = 20
```

```
400 rmat(1,1) = 900
410 increase = 9   ! # cf consecutive ellipse area increases before
                     increasing q.
420 decrease = 5   ! # cf consec. ell area decreases before dec. q.
430 cl = 0
440 up = 0
450 down = 0
460 upflag = 0
470 downflag = 0
480 count = 0
490 out = 0
500 llarge = 0
510 mat ssum = zer
520 mat sumsq = zer
530 mat prod = zer
540 assign# 2 tc "et2set, drive0"
550 assign# 3 to "perr8 .storag"
560 assicn# 4 tc "statmat8a .storag"
570 assign# 5 to "errex+8a .storag"
580 assign# 6 tc "xhatex8a .storag"
590 assign# 7 to "noitrak8 .storag"
600 mat imat = idn
610 mat reset = (100)*imat
620 mat pk1k = reset
630 read# 1 ; xhk1k()
640 read# 2 ; phi()
650 mat phit = trn(phi)
660 read# 2 ; gama()
670 mat gamat = trn(gama)
680 hmat(1,2) = 0
690 hmat(1,4) = 0
700 covw(1,2) = 0
710 covw(2,1) = 0
720 for k = 1 tc high
730 !********** reinitialize if conditions are met *****************
740   if upflag=1 then covw(1,1)=2*covw(1, 1) @ upflag=0
750   if downflag=1 then ccvw (1,1)=.5*covw (1,1) @ downflag=0
760   ccvw(2,2)=covw(1,1)
770   if statplt = 1 then 2480
780   mat tempq = covw*camat
790   mat gmat = gama*tempq
800   read# 1 ; trak()
```

```
810   if noitrak = 1 then read# 7 ; zmat() @ goto 850
820   v(1) = noise*2*(rnd - .5)
830   zmat(1,1) = sqr(trak(1,1)**2 + trak(3,1)**2) + v(1)
840   print# 7 ; zmat()
850  !*****************conduct 3-sigma gate test *********************
860   maxpk = pk1k(1,1)
870   for i = 2 to 4
880       if pk1k(i,i)>maxpk then maxpk = pk1k(i,i)
890   next i
900   gate = 3*sqr(maxpk + rmat(1,1))
910   hatrak = sqr(xhk1k(1,1)**2 + xhk1k(3,1)**2)
920   diff = abs(hatrak - zmat(1,1))
930   if diff<gate then 980
940   mat pkk = pk1k
950   mat xhat = xhk1k
960   print# 6 ; xhat()
970   goto 1210
980   l = k-1
990   hmat(1,1) = xhk1k(1,1)/sqr(xhk1k(1,1)**2 + xhk1k(3,1)**2)
1000  hmat(1,3) = xhk1k(3,1)/sqr(xhk1k(1,1)**2 + xhk1k(3,1)**2)
1010  mat hmatt = trn(hmat)
1020  mat temp1 = pk1k*hmatt
1030  mat temp2 = hmat*temp1
1040  mat temp3 = temp2 + rmat
1050  mat temp3in = (1)/temp3
1060  mat temp4 = hmatt*temp3in
1070  mat gain = pk1k*temp4
1080 !**************compute the covariance matrix p(k/k)**************
1100  mat temp5 = gain*hmat
1110  mat temp6 = imat - temp5
1120  mat pkk = temp6*pk1k
1140 !************* compute the estimate xhat(k/k) *******************
1150  chk1k(1,1) = sqr(xhk1k(1,1)**2 + xhk1k(3,1)**2)
1160  mat temp7 = zmat - chk1k
1170  mat temp8 = gain*temp7
1180  mat xhat = xhk1k + temp8
1190  print# 6 ; xhat()
1210 !****************** compute xhat(k+1/k) ************************
1220  xhk1k(1,1) = xhat(1,1) + xhat(2,1)
1230  xhk1k(2,1) = xhat(2,1)
1240  xhk1k(3,1) = xhat(3,1) + xhat(4,1)
1250  xhk1k(4,1) = xhat(4,1)
```

```
1270 !****************** compute p(k+1/k) ****************************
1280  mat temp9 = pkk*phit
1290  mat temp9a = phi*temp9
1300  mat pk1k = temp9a + qmat
1310  if k<start then goto 3370
1320  if pplot = 0 then 1370
1330  perr(1) = pkk(s,s)
1340  perr(2) = pkk(t,t)
1350  perr(3) = pkk(s,t)
1360  print# 3 ; perr()
1370 !****************  compute statistics **************************
1390  mat err = trak - xhat
1400  if window > 0 then 1760
1410  if norm<> 1 then 1470
1420  for i = 1 to 4
1430      if trak(i,1) = 0 then trak(i,1) = 1
1440  next i
1450  mat err = err/trak
1460  mat ssum = err + ssum
1470  print# 5 ; err()
1480 !***************** compute the mean of the error ***************
1490  mat mean = (1/k)*ssum
1500 !***************** compute the variance of the error ************
1510  mat errsq = err.err
1520  mat sumsq = errsq + sumsq
1530  mat meansq = (1/k)*sumsq
1540  mat meanksq = mean.mean
1550  mat var = meansq - meanksq
1560  !******** compute the running product mean ********************
1570  for i = 2 to n
1580      for j = 1 to i-1
1590          tprod(i,j) = err(i)*err(j)
1600          prod(i,j) = tprod(i,j) + prod(i,j)
1610          mnprod(i,j) = prod(i,j)/k
1620      next j
1630  next i
1640 !******** compute the diagonal terms of the cov of error matrix**
1650  for i = 1 to n
1660      covmat(i,i) = var(i)
1670  next i
1680 !****** compute the off-diagonal terms of cov of error matrix ***
1690  for i = 2 to n
```

```
1700        for j = 1 to i-1
1710             covmat(i,j) = mnprod(i,j) - mean(i)*mean(j)
1720             ccvmat(j,i) = covmat(i,j)
1730        next j
1740   next i
1750   goto 2490
1760   fcr i = 1 to 4
1770        werr(i,k) = err(I)
1780   next i
1790   if k = 1 then 1850
1800   if k>window then 2170
1810   fcr i = 1 to 4
1820        wssum(i,k) = werr(i,k) + wssum(i,k-1)
1830   next i
1840   goto 1880
1850   fcr i = 1 to 4
1860        wssum(i,k) = werr(i,k)
1870   next i
1880   !*************** compute the mean of the error *****************
1890   fcr i = 1 to 4
1900        wmean(i,k) = (1/k)*wssum(i,k)
1910   next i
1920  !***************** ccmpute the variance cf the error ************
1930   fcr i = 1 to 4
1940        werrsq(i,k) = werr(i,k)*werr(i,k)
1950   next i
1960   if k = 1 then 2010
1970   fcr i = 1 to 4
1980        wsumsq(i,k) = werrsq(i,k) + wsumsq(i,k-1)
1990   next i
2000   goto 2040
2010   fcr i = 1 to 4
2020        wsumsq(i,k) = werrsq(i,k)
2030   next i
2040   for i = 1 to 4
2050        wmeansq(i,k) = (1/k)*wsumsq(i,k)
2060        wmeanksq(i,k) = wmean(i,k)*wmean(i,k)
2070        wvar(i,k) = wmeansq(i,k) - wmeanksq(i,k)
2080   next i
2090   !************* ccmpute the running product mean ***************
2100   wtprod(k) = werr(s,k)*werr(t,k)
2110   if k = 1 then 2140
```

```
2120  wprod(k)  = wtprod(k)  + wprod(k-1)
2130  goto 2150
2140  wprod(k)  = wtprod(k)
2150  wmnprod(k) =  (1/k)*wprod(k)
2160  goto 2390
2170 !**********  compute the window statistics *********************
2180  for i = 1 to 4
2190       wssum(i,k) = werr(i,k) + wssum(i,k-1)  - werr(i,k-window)
2200  next i
2210 !************ compute the mean of the error *******************
2220  for i = 1 to 4
2230       wmean(i,k)  = (1/window)*wssum(i, k)
2240  next i
2250 !************* compute the variance of the error ***************
2260  for i = 1 to 4
2270       werrsq(i,k) = werr(i,k)*werr(i,k)
2280       wsumsq(i,k) = werrsq(i,k)+wsumsq(i,k-1)-werrsq(i,k-window)
2290  next i
2300  for i = 1 to 4
2310       wmeansq(i,k) = (1/window)*wsumsq(i,k)
2320       wmeanksq(i,k) = wmean(i,k)*wmean(i,k)
2330       wvar(i,k)  = wmeansq(i,k)  - wmeanksq(i,k)
2340  next i
2350 !************ compute the running product mean *****************
2360  wtprod(k) = werr(s,k)*werr(t,k)
2370  wprod(k)  = wtprod(k)  + wprod(k-1)  -wtprod(k-window)
2380  wmnprod(k) =  (1/window)*wprod(k)
2390  covmat(s,s)  = wvar(s,k)
2400  covmat(t,t)  = wvar(t,k)
2410  covmat(s,t)  = wmnprod(k)  - wmean(s,k)*wmean(t, k)
2420  ccov(k)  = covmat(s,s)*covmat(t,t)
2430  if k  = 1 then 2480
2440  if ccov(k)>ccov(k-1)  then up = up + 1 @ down = 0
2450  if ccov(k)<ccov(k-1)  then down = down + 1 @ up = 0
2460  if up = increase then upflag = 1 @ up = 0
2470  if down = decrease then downflag = 1 @ down = 0
2480  if statplt = 1 then read# 4 ; covmat(s,s),covmat(t,t),
              covmat(s,t) @ goto 2500
2490  print# 4 ; covmat(s,s),covmat(t,t),covmat(s,t)
2500 !********************plot error ellipse **************************
2510  if k<first then 3370
2520  if k = first or (k-first) mod nr = 0 then 2530 else 3370
```

```
2530  count = count + 1
2540  if pplot = 1 then 2590
2550   ekk(1) = covmat(s,s)
2560   ekk(2) = covmat(t,t)
2570   ekk(3) = covmat(s,t)
2580  goto 2600
2590  mat ekk = perr
2600  lx = ekk(1)
2610  ly = ekk(2)
2620  mmax = max(lx,ly)
2630  mmin = min(lx,ly)
2640  if mmin<0 then 3290
2650  if mmax/mmin<10000 then 2670
2660  if mmax=lx then llarge = 1 else llarge = 2
2670  if ekk(3)<>0 then 2700
2680    theta = 0
2690    goto 2770
2700  if ekk(1)<> ekk(2) then 2730
2710    theta = 0
2720    goto 2770
2730    theta = .5*atn(2*ekk(3)/ekk(1)-ekk(2))
2740    lx =  (ekk(1) + ekk(2))/2 + ekk(3)/sin(2*theta)
2750    ly =  (ekk(1) + ekk(2))/2 - ekk(3)/sin(2*theta)
2760  if lx<0 or ly<0 then 3370
2770  a = sqr(lx)
2780  b = sqr(ly)
2790  area = 3.1416*a*b
2800  if pltarea = 1 then 3360
2810  if k = first then 2840
2820  if count = 1 and sscale = 1 then 2840
2830  goto 2880
2840  smax = max(a,b)
2850  if smax<.000001 then 3290
2860  pscale=2*smax
2870  nscale = -(2*smax)
2880  if pl = 1 then plotter is 1 else plotter is 805
2890  if cl <> 0 then 2930
2900  pt = chg - .5
2910  gclear
2920  if pl = 0 then limit 28,241,35,184
2930  if pl = 1 then locate 25,145,16,90 else locate 30,108,18,96
2940  scale nscale,pscale,nscale,pscale
```

```
2950  if cl <> 0 then 3020
2960  move .8*pscale,.1*pscale
2970  label "x-position (ft)"
2980  move .2*pscale,1.05*pscale
2990  label "y-position (ft)"
3000  move .9*pscale,1.05*pscale
3010  label "time";"  area (sq ft)";"  legend"
3020  if sscale <>1 and k <> first then 3050
3030  if cl = 0 then laxes pscale/2,pscale/2,0,0
3040  goto 3060
3050  if pl <> 1 and cl = 0 then laxes pscale/2,pscale/2,0,0
3060  move .9*pscale,pt/chg*pscale
3070  if llarge = 0 then 3120
3080  if llarge=1 then label using 3090; "  ";k;"  ";area;"  x-axis"
      else label using 3090; "  ";k;"  ";area;"  y-axis"
3090  image 2a,dd,2a,4de,7a
3100  llarge = 0
3110  goto 3170
3120  if count = 1 then label using 3160; "  ";k;"  ";area;" ----"
3130  if count = 2 then label using 3160; "  ";k;"  ";area;" ...."
3140  if count = 3 then label using 3160; "  ";k;"  ";area;"- - -"
3150  if count = 4 then label using 3160; "  ";k;"  ";area;"_ _ _"
3160  image 2a,dd,2a,4de,6a
3170  move 0,0
3180  if count = 2 then line type 3
3190  if count = 3 then line type 5
3200  if count = 4 then line type 1
3210  for ang = 0 to 360 step stp
3220     xx = cos(ang)*a
3230     yy = sin(ang)*b
3240     x1 = xx*cos(theta) - yy*sin(theta)
3250     y1 = yy*cos(theta) + xx*sin(theta)
3260     rplot x1,y1
3270  next ang
3280  goto 3310
3290  disp "no ellipse plotted, out of scale for k=";k
3300  goto 3370
3310  cl = cl + 1
3320  pt = pt - .4
3330  if cl = chg then cl = 0
3340  if count = chg then count = 0
3350  goto 3370
```

```
3360  disp "k=";k;area
3370 next k
3380 assign# 1 to *
3390 assign# 2 to *
3400 assign# 3 to *
3410 assign# 4 to *
3420 assign# 5 to *
3430 assign# 6 to *
3440 assign# 7 to *
3450 end
```

LIST OF REFERENCES

1. Sorenson, H.W., "Kalman Filtering Techniques", Advances in Control Systems, v. 3, pp. 219-292, Academic Press, 1966.

2. Kirk, D.E., EE4413 Class Notes "Optimal Estimation: An Introduction to the Theory and Applications", (unpublished), Naval Postgraduate School, 1975.

3. Jazwinski, A.H., Stochastic Processes and Filtering Theory, Academic Press, 1970.

4. Cramer, H., Mathematical Methods of Statistics, Princeton University Press, 1961.

5. Mitschang, G.W., An Application of Nonlinear Filtering Theory to Passive Target Location and Tracking, PhD Dissertation, Naval Postgraduate School, 1974.

6. Heffes, H., "The Effects of Erroneous Models on the Kalman Filter Response", IEEE Trans. Automatic Control, v. 11, pp. 541-543, 1966.

7. Nishimura, T., "Error Bounds of Continuous Kalman Filters and the Application of Orbit Determination Problems", IEEE Trans. Automatic Control, v. 12, pp. 268-275, 1967.

BIBLIOGRAPHY

Anderson, B.D., and Moore, J.B., Optimal Filtering, Prentice Hall, 1979.

Meditch, J.S., Stochastic Optimal Linear Estimation and Control, McGraw-Hill, 1969.

Sage, A.P., and Melsa, J.L., Estimation Theory with Applications to Communications and Control, McGraw-Hill, 1971.

Ladas, J.E., Interactive Monte Carlo Simulation Program for Evaluation of Estimators and Predictors for Fire Control Systems, Master's Thesis, Naval Postgraduate School, 1981.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
4.	Professor A. Gerba, Jr., Code 62Gz Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
5.	Professor H. Titus, Code 62Ts Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
6.	Commanding Officer Naval Underwater Weapons Engineering Station Keyport, Washington 98345	1
7.	Commander Joseph Jaros, USN Command and Control Engineering Center Code G520 Reston, Virginia 22090	2

END

FILMED

[illegible]-84

DTIC